Министерство просвещения РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра информатики, информационных технологий
и методики обучения информатике

Разработка сервиса школьного онлайн-тестирования для мобильных устройств

Выпускная квалификационная работа
по направлению подготовки «44.03.01 Педагогическое образование», профиль «Информатика»

Исполнитель: студент группы ИНФ-1601z
Поташкин А.И.
Руководитель: старший преподаватель
кафедры ИИТиМОИ
Алексеевский П.И.

Работа допущена к защите
«____»_____ 2021 г.

Руководитель _____

Екатеринбург – 2021

РЕФЕРАТ

Поташкин А.И. РАЗРАБОТКА СЕРВИСА ШКОЛЬНОГО ОНЛАЙН-ТЕСТИРОВАНИЯ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ, выпускная квалификационная работа: 71 стр., рис. 30, библ. 27 назв., приложений 1 + приложения на DVD

Ключевые слова: ПЕДАГОГИЧЕСКОЕ ТЕСТИРОВАНИЕ, ВЕБ-ПРИЛОЖЕНИЕ, ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ.

Объект исследования – педагогическое тестирование в школах.

Предмет исследования – использование сервисов онлайн-тестирования на мобильных устройствах как способ автоматизации педагогического тестирования в школах.

Цель работы – разработка сервиса школьного онлайн-тестирования для мобильных устройств, позволяющего автоматизировать педагогическое тестирование, ускорить проведение педагогической диагностики за счет использования мобильных устройств.

В работе описаны результаты анализа доступных веб-решений для школьного онлайн тестировано, теоретические аспекты педагогического тестирования и технологий разработки веб-приложения, проектирования и программной реализации веб-приложения, построенного по модели MVC.

Исходный код веб-приложения написан на языке PHP, для работы с СУБД используется база данных MySQL.

Проведена апробация, разработанного веб-приложения, методом экспертных оценок. Экспертами выступали 9 студентов Уральского государственного педагогического университета.

Оглавление

Введение

В век информационных технологий всё чаще встает вопрос о разработке и внедрении в жизнедеятельность человека разнообразных приложений, упрощающих его жизнь и автоматизирующих его повседневные процессы.

На данный момент, практически у любого человека есть мобильное устройство, с установленными приложениями для решения различных видов задач. От широко распространённых до узкоспециализированных.

Использование программного обеспечения для автоматизации педагогических процессов позволяет анализировать данные в режиме реального времени, вести активное взаимодействие с потребителями, в данном случае, обучающимися, в том числе упрощает процессы педагогической деятельности, увеличивает число доступных педагогических методов. А использование веб-приложений позволяет не зависеть от определенной операционной системы или платформы.

Таким образом, для облегчения повседневной рабочей деятельности преподавателя, нужно разрабатывать программное обеспечение и внедрять его в рабочий процесс.

Для решения задачи данной выпускной квалификационной был выбран текстовый редактор Sublime Text 3, локальный сервер Денвер 3 и язык программирования PHP.

Разрабатываемое веб-приложение направлено на автоматизацию педагогического тестирования, ускорения проведения педагогической диагностики. Внедрение разрабатываемого ПО позволит сократить ручную работу преподавателя на проверку результатов школьного тестирования.

Первый раздел выпускной квалификационной работы посвящен анализу решений на рынке онлайн-тестирования и рассмотрению понятий «педагогическая диагностика», «педагогический тест», даются описание и примеры тестовых заданий. Также в разделе рассматриваются теоретические

аспекты моделей жизненного цикла ПО, шаблонов проектирования ПО и технологий разработки веб-приложений.

Во втором разделе описывается структура разрабатываемого веб-приложения и используемых элементов, основные методы веб-приложения и инструкция пользователю педагогического программного средства.

Объектом исследования является педагогическое тестирование в школах.

Предметом исследования – использование сервисов онлайн-тестирования на мобильных устройствах как способ автоматизации педагогического тестирования в школах.

Целью выпускной квалификационной работы является разработка сервиса школьного онлайн-тестирования для мобильных устройств, позволяющего автоматизировать педагогическое тестирование, ускорить проведение педагогической диагностики за счет использования мобильных устройств.

Задачи выпускной квалификационной работы:

1. Провести анализ решений онлайн-тестирования и определить;
2. Изучить основные понятия аспектов педагогического тестирования, моделей жизненного цикла ПО и технологии используемые в процессе разработки веб-приложения;
3. Освоить навыки создания веб-приложений с MVC архитектурой;
4. Создать веб-приложение для школьного онлайн-тестирования с использование текстового редактора Sublime Text 3, локального сервер Денвер 3 и языка программирования PHP;
5. Опубликовать веб-приложение в сети Интернет.
6. Провести апробацию веб-приложения школьного онлайн-тестирования методом экспертных оценок.

Глава 1. Теоретическая часть

1.1 Анализ решений на рынке онлайн-тестирования

На рынке компьютерного тестового контроля, которые позволяют проводить онлайн-тестирования обучающихся, множество решений для разных типов педагогических задач. От веб-решений для создания тестов или опросов, для которых педагогическое направление не является основным, до систем управления обучением (LMS – learning management system) – программных решений как для администрирования учебных курсов в рамках дистанционного обучения [17], так и для проведения онлайн-тестирования в частности. Давайте рассмотрим некоторые из них, и попробуем выделить моменты, которые могут помешать реализовать онлайн-тестирование с использованием мобильных устройств:

КонструкторТестов.ру (konstruktortestov.ru) – бесплатное веб-решения создания и прохождения тестов. На нём имеется отдельный сервис для преподавателей, где возможно создания тестов для различных классов,

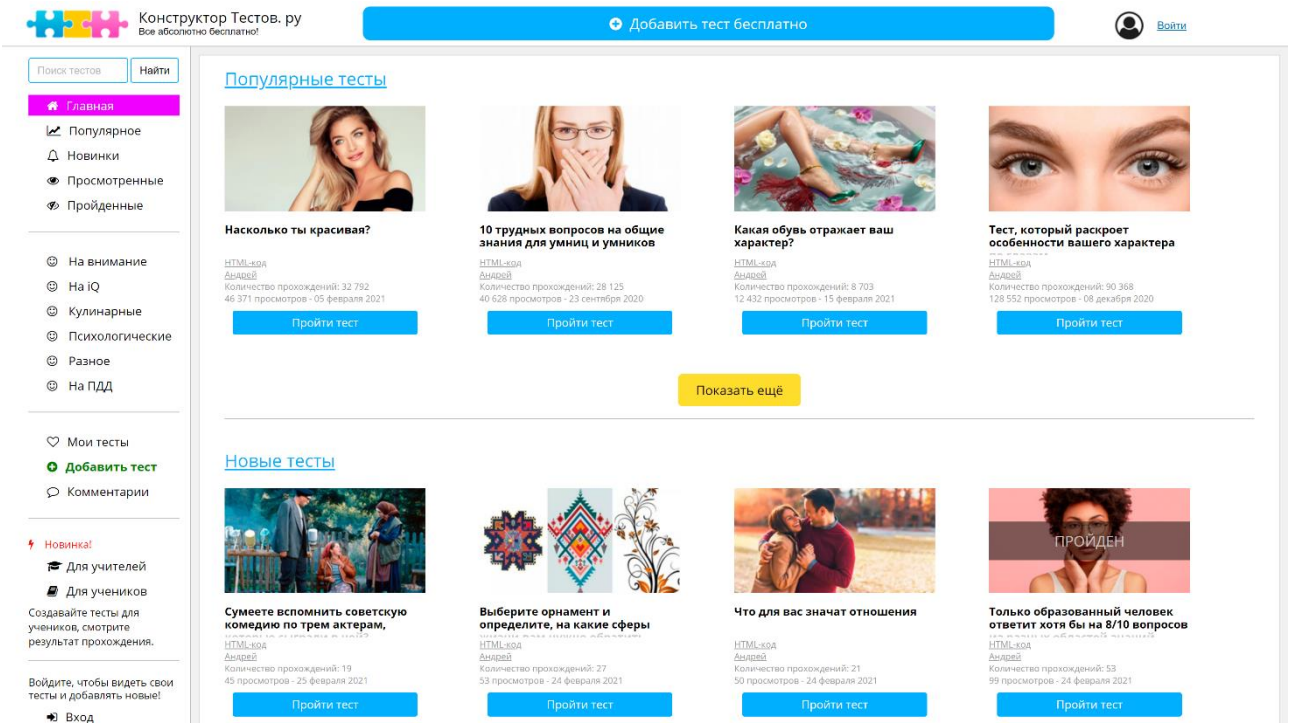


Рис.1. Скриншот стартовой страницы konstruktortestov.ru [3]

получения статистики прохождения, и выставление запрета на повторное прохождение.

Недостатки: для того чтобы учащиеся могли пройти тест, учителю, предварительно, необходимо создать список учеников, и предоставить каждому ID и пароль для входа. Также из доступных форм тестовых заданий имеется только задания закрытого типа с одиночным/множественным выбором (рис.2).

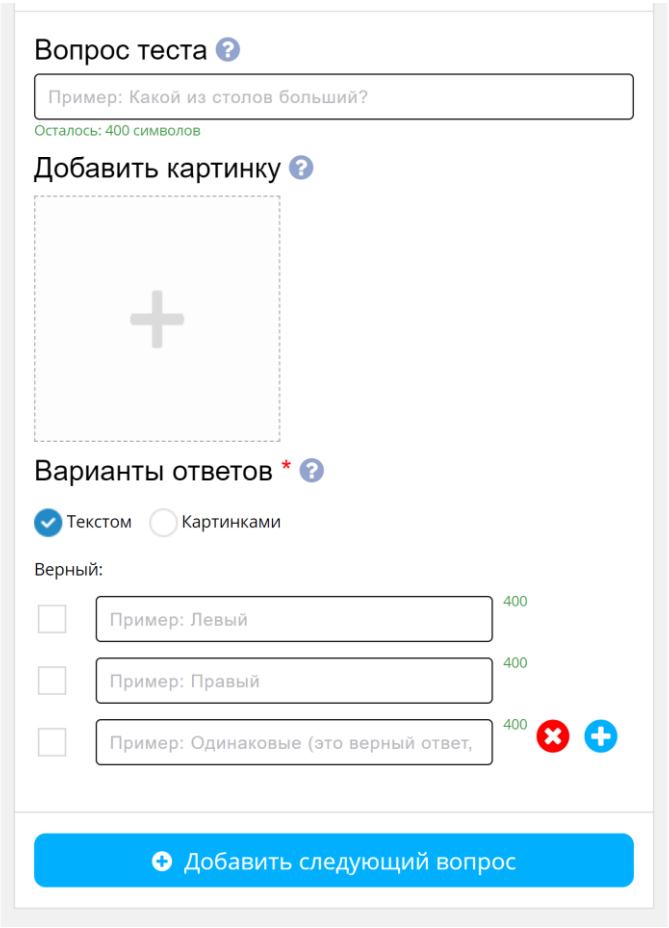


Рис.2. Скриншот страницы создания теста konstruktortestov.ru [3]

LearningApps.org (learningapps.org) – бесплатный веб-ресурс, созданный для поддержки обучения и преподавания с помощью небольших общедоступных интерактивных упражнений [24]. Преподаватель может создавать «упражнения»

из большего количества вариантов тестовых форм, и давать доступ к прохождению теста по ссылке, регистрироваться обучающимся не нужно.

Рис.3. Скриншот стартовой страницы теста learningapps.org [24]

Недостатки: веб-сайт не ориентирован для использования на мобильных устройствах.

INDIGO (indigotech.ru) – профессиональный инструмент автоматизации процесса тестирования и обработки результатов, предназначенный для решения

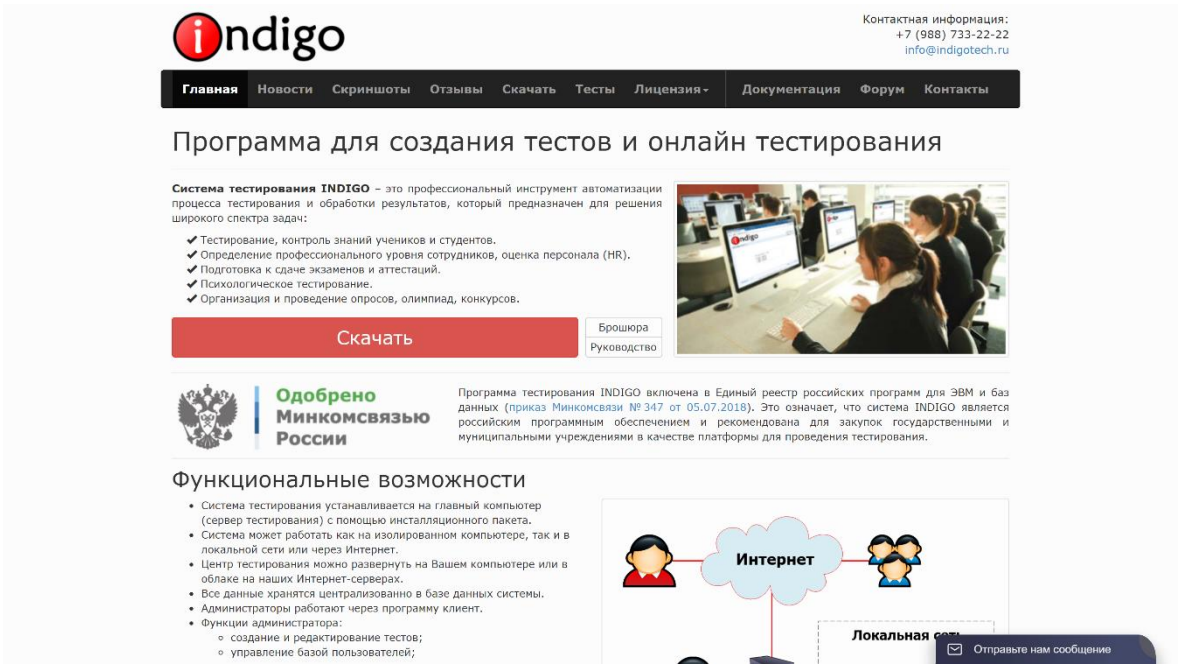


Рис.4. Скриншот стартовой страницы indigotech.ru [23]

широкого спектра задач [23]. Данная информационная система позволяет создавать и редактировать тесты, администрировать пользователей, устанавливать правила тестирования, а также экспортировать результаты в различные форматы. Тестирование происходит в браузере.

Недостатки: информационная система является клиент-серверной в полной её мере, для её развертывания необходимо выделять отдельный локальный физический сервер или пользоваться платным облачным решением компании разработчика. Обучающихся необходимо предварительно добавлять в базу данных, либо им придется регистрироваться самостоятельно. Доступ к тесту предоставляется по логину/паролю. Хоть в достоинствах ИС, разработчик заявляет поддержку браузеров мобильных устройств, прохождение тестов на

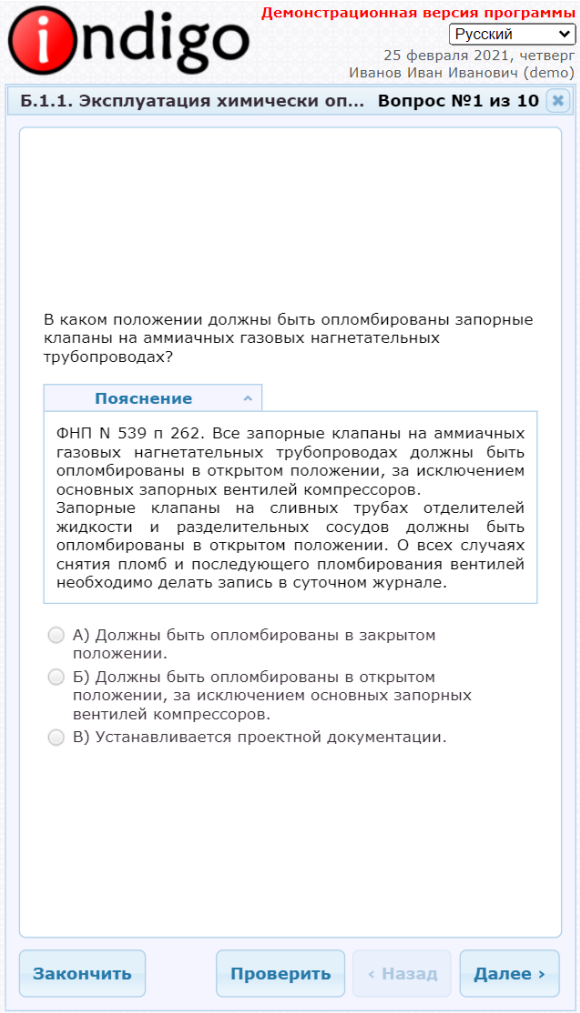


Рис.5. Скриншот примера тестового задания indigotech.ru [23]

маленьких экранах смартфонов – не является удобным (рис.5). Ко всему прочему, решение является платны.

Moodle (moodle.org) – свободно распространяющееся LMS-решение с открытым исходных кодом, предоставляющее возможность создавать сайты для онлайн-обучения. На платформе, также, существует возможность проводить онлайн-тестирование.

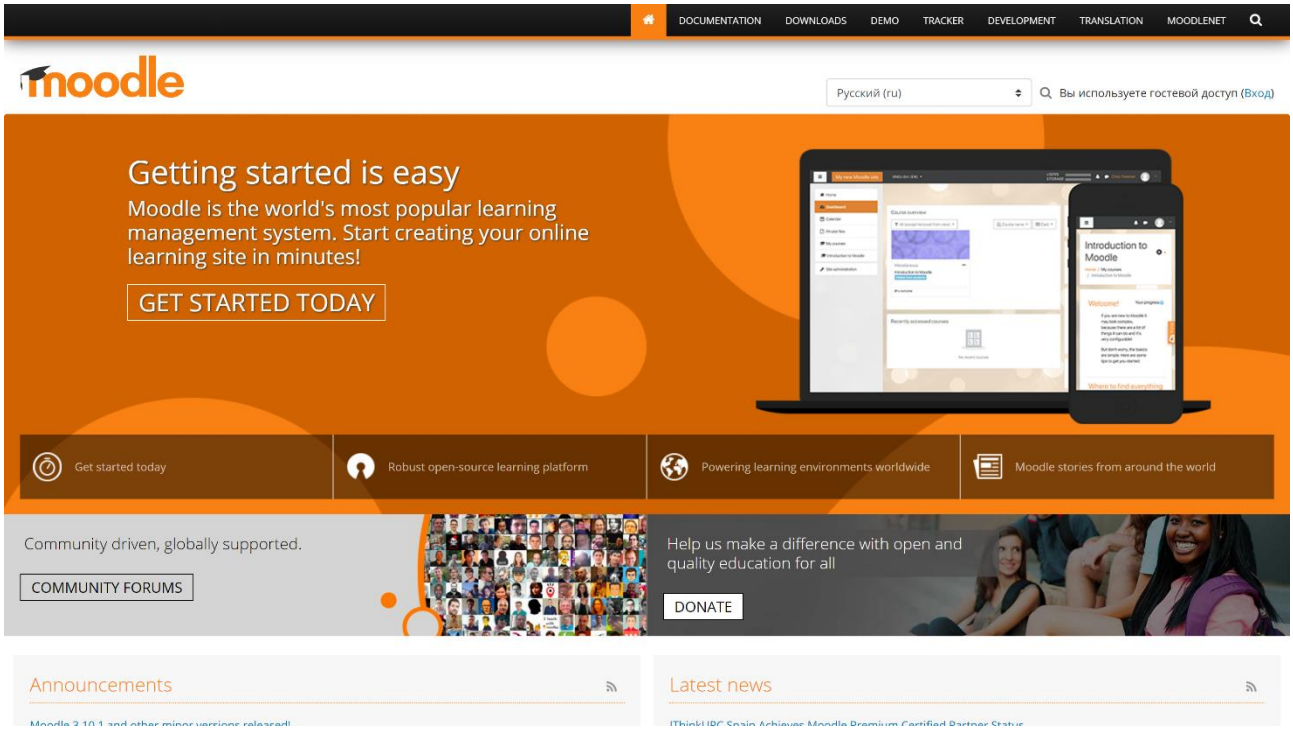


Рис.6. Скриншот стартовой страницы moodle.org [25]

Недостатки: как и в случае с INDIGO, необходимо разворачивать серверную составляющую решения. К тому же, в данном случае сервер должен быть на UNIX-подобной ОС. Само решение обладает огромным функционалом, что не всегда является плюсом для реализации моментального онлайн-тестирования. Доступ к тестам предоставляется для заранее введённых/зарегистрированных обучающихся. Для работы с системой, обучающимся предлагают скачивать мобильные приложения под разные платформы.

Таким образом, рассмотрев некоторые веб-решения, можно обозначить важные моменты в будущем веб-приложении, позволяющие преподавателю осуществить онлайн-тестирования с помощью мобильных устройств:

- серверная составляющая сервиса, уже должна быть развернута разработчиком на собственных серверах;
- доступ к тесту необходимо реализовать так, чтобы не приходилось предварительно составлять списки обучающихся, создавать им учетные записи;
- для доступа к тесту необходимо использовать ссылки или, например, коды доступа к тесту;
- сервис должен работать в браузере мобильного устройства, адаптироваться под различные диагонали экранов;
- в сервисе должны быть реализованы все формы тестовых заданий.

1.2 Педагогическая тестирование и формы тестовых заданий

Диагностика рассматривается как область научных знаний, с одной стороны, и как область практики, с другой. Термин «диагностика» активно используется в педагогике, технике, медицине, психологии, и других областях науки и социальной практики. Сегодня существуют разные трактовки понятия «диагностика».

В переводе с греческого «диагностика» (греч. diagnosis) – означает «различительное познание». С позиции общей методологии (науки об организации деятельности) диагностика может рассматриваться как специализированная область познания, включающая теорию и методы организации процессов распознавания, а также принципы организации и построения средств диагноза [19]. В узком смысле под диагностикой понимают процесс распознавания состояния определенного объекта или системы путем регистрации его существенных параметров и последующего отнесения к

определенной диагностической категории с целью прогноза его поведения, а также принятия решения о возможностях воздействия на это поведение [2].

Диагностика занимает особое место в процессах управления, поскольку обеспечивает информационное наполнение канала обратной связи, на основании которого принимаются управленческие решения. Каждая мера контроля должна начинаться со сбора диагностической информации и заканчиваться повторным диагностическим обследованием (для сравнения желаемых и фактических результатов воздействия).

Понятие «педагогическая диагностика» было введено К. Ингенкампом в 1968 году. Педагогическую диагностику также принято рассматривать в узком смысле как область практической деятельности, направленной на оптимизацию педагогического процесса [19]. Как область познания педагогическая диагностика – это теоретико-прикладная отрасль педагогики, изучающая закономерности вынесения диагностических суждений о разнообразных элементах и параметрах педагогических систем и отношений, правила проведения диагностических процедур; принципы, методы и формы диагностики. В широком смысле сущность педагогической диагностики – в изучении актуального состояния разнообразных элементов и параметров педагогической системы с целью оптимального решения педагогических задач [16].

- Целью образовательной диагностики является постановка диагноза, который формирует основу системы мер, которые можно использовать для оказания быстрой и эффективной помощи ученику, учителю и директору школы.
- Задача диагностики состоит в распознавании существа и состояния изучаемых объектов по совокупности устойчиво наблюдаемых признаков.
- Объектом педагогической диагностики является учебно-воспитательный процесс и его результаты.

- Предметом педагогической диагностики являются какие-либо подсистемы объекта.
- Результат педагогической диагностики – педагогический диагноз.

Диагностика – обязательная составляющая деятельности учителя, а умение правильно использовать диагностические процедуры – часть профессиональной культуры учителя. Диагностическая деятельность учителя направлена на изучение индивидуально-психологических особенностей ученика и группы учеников, выявление и оценку их знаний и умений по учебной дисциплине.

На практике, диагностика проводится для получения информации об объекте, на основании которой строится прогноз его поведения и выбирается метод последующего воздействия. Успех диагностики во многом определяется выбором методов и средств, соответствующих практической задаче.

Одним из оснований классификации методов диагностики может быть мера «субъективности-объективности», которой обладают результаты диагностики. В случае применения объективных методов влияние диагноста на результаты минимально. При использовании субъективных методов результаты диагностики напрямую зависят от компетентности диагноста. С учетом двух признаков – «субъективности-объективности» и операционально-технологической составляющей можно выделить малоформализованные (экспертные) и формализованные (измерительные) методы диагностики. К малоформализованным методам относят наблюдение, беседу, интервью, анализ результатов деятельности, которые допускают некоторую формализацию (алгоритмизацию) процедуры диагностики, фиксирования и анализа ее результатов. К формализованным (часто их называют стандартизированными) методам относят анкетирование и тестирование [19]. Подробно остановимся на формализованных методах, в частности, на тестировании.

Тестирование – научно-обоснованный процесс измерения интересующих исследователя свойств личности [19]. Метод тестирования широко применяется

в педагогической практике. Основная цель педагогического теста – выявить и измерить знания и умения учеников или студентов в определенной области содержания учебной дисциплины (знание темы, раздела, дисциплины в целом). К. Ингенкамп предложил рассматривать метод тестирования как метод педагогической диагностики, «с помощью которого выборка поведения, репрезентирующая предпосылки или результаты учебного процесса, должна максимально отвечать принципам сопоставимости, объективности, надежности и валидности измерения, должна пройти обработку и интерпретацию и быть готовой к использованию в педагогической практике» [6].

Тесты используются на разных этапах изучения учебной дисциплины: на входном, текущем (оперативном), тематическом, рубежном и итоговом контроле. В последние годы особый интерес для учителей вызывают тесты для адаптивного контроля. Основная идея такого контроля – выбор заданий, соответствующих актуальной и ближайшей зоне развития обучающегося, а также задач, находящихся за пределами зоны ближайшего развития и непосильных для него на данном этапе обучения. Решение такой проблемы возможно с помощью математических моделей современной теории тестирования.

Уровень знаний обучающихся и уровень трудности заданий измеряются в единой шкале (шкале логитов) [19]. Анализ различий этих уровней позволяет выделить задания, по сложности, соответствующие зонам актуального и непосредственного развития ученика. Если уровень знаний выше, чем уровень сложности предложенного задания, ученик, скорее всего, выполнит задание без посторонней помощи. Такие задачи можно отнести к области ближайшего развития. В противном случае (сложность задания выше уровня знаний ученика) задание относится к зоне ближайшего развития. Практическая реализация идеи адаптивного контроля затруднена тем, что необходимо использовать

специальные компьютерные программы, реализующие адаптивные алгоритмы тестирования, основанные на моделях латентно-структурного анализа [22].

Использование компьютерных технологий дает возможность автоматизировать процедуру опроса и оценки ответа, а также обработки результатов. Однако используемые алгоритмы не являются чем-то принципиально новым – они построены или являются обобщением сформулированных универсальных положений и относятся к обучающим тестам для любой технологии сбора первичной информации и ее последующей обработки.

Для диагностики успеваемости обучающихся по дисциплине используются различные средства и методы. Как упоминалось выше, педагогический тест в настоящее время является популярным инструментом измерения в этой области. В научной-педагогической литературе и нормативных документах можно найти значительное количество определений теста. Рассмотрим определение, данное В.С. Аванесовым:

Педагогический тест – система заданий возрастающей трудности специфической формы, позволяющая выявить и измерить качество, уровень и структуру знаний испытуемого в определенной области содержания [6].

Из определения, можно сформулировать основную задачу тестирования – дифференциацию обучаемых по уровню подготовленности.

Применение метода педагогического тестирования в педагогической диагностике связано с его достоинствами:

- более высокая объективность оценки уровня учебных достижений обучаемого за счет: а) стандартизации процедуры опроса; б) отображения в тесте значительного объема проверяемого учебного материала; в) применения статистических методов обработки результатов тестирования;

- тестирование является педагогической технологией, отвечающей всем признакам технологии (гарантированное достижение результата, переносимость, возможность автоматизации процедуры);
- фронтальный характер опроса (массовость) – с помощью одного измерителя выявляется уровень учебных достижений всех (неограниченного числа) учащихся, что обеспечивает сопоставимость результатов и выявление индивидуальных и групповых затруднений;
- быстрота получения результатов (оперативность проверки);
- возможность совершенствования теста как средства диагностики за счет количественной оценки результатов тестирования [19].

Педагогические тесты можно классифицировать

- по интерпретации результатов тестирования:
  - критериально-ориентированные;
  - нормативно-ориентированные.
- по уровням контроля:
  - тест для входного контроля;
  - тест для текущего (оперативного) контроля;
  - тест для итогового оценивания (суммирующего).
- по предметной «чистоте» (отображение проверяемого содержания):
  - гомогенные (касаются содержания отдельной учебной дисциплины);
  - гетерогенные (по совокупности дисциплин);
  - интегративные (ответы на поставленные вопросы требуют знания учебного материала по двум и более дисциплинам).
- по форме предъявления материала и форм сбора первичных данных:
  - бланковые тесты (на бумажном носителе);

    - компьютерные тесты (на электронном носителе).
- по уровню разработки:
    - неформальные (учительские, преподавательские);
    - стандартизированные (профессиональные).

Существуют следующие формы тестовых заданий:

- задания одиночного выбора;
- задания множественного выбора;
- задания открытого типа;
- задания соответствия;
- задания на установления последовательности.

Рассмотрим каждую из форм подробнее.

Задания одиночного выбора – это своего рода «атом» конструкции тестового задания, поскольку все остальные задания закрытого типа, в конечном счете, могут быть сведены к их совокупности – выбор может быть организован по-разному – этим определяется вероятность угадывания ответа и сложность концептов, необходимых для выполнения задания.

Структура простейшего тестового задания включает формулировку вопроса и нескольких (минимум – двух) вариантов ответов, из которых один обязательно и однозначно является верным, а остальные – однозначно нет. К заданию всегда прилагается «ключ» – правильный ответ, из сравнения с которым делается заключение о правильности выполнения задания: если ответ тестируемого совпадает с эталоном, утверждение становится истинным (True, 1), если нет – утверждение ложно (False, 0) [19].

Пример:

Русский писатель А.П. Чехов родился в…

1. 1834 г.

2. **1860 г.**
3. 1849 г.
4. 1871 г.

Правильным, является ответ «2», а остальные варианты выступают в роли дистракторов.

Дистрактор – (от англ. distract – отвлечение внимания) предлагаемый в тестовых заданиях выбора альтернативный вариант ответа, не являющийся истинным, но внешне близкий к нему [19]. Для корректности задания, к дистракторам применяются следующие требования:

- однозначность – все варианты ответа задания должны иметь однозначную интерпретацию и не допускать никакой другой интерпретации, которая делает их верными при определенных условиях (возможно, не предусмотренных автором теста);
- правдоподобность – в них не должно быть явно видимых ложных утверждений (например, если один из ответов в данном примере был 1921 или 1678);
- смысловая корректность – если дистракторами являются формулировки правил, законов и т. д. (в текстовом или формульном представлении), то не должно быть заведомо неправильных формулировок (например, «сила тока в цепи прямо пропорциональна сопротивлению и обратно пропорциональная напряжению»); в качестве дистракторов необходимо выбирать утверждения, истинные по своему смыслу, но не имеющие отношения к поставленному вопросу.

Легко оценить вероятность угадывания ответа в заданиях с одиночным выбором. Если общее количество вариантов ответов n, то, очевидно,

$$P = \frac{1}{n}$$

Обычно в тестовых заданиях такого типа n составляет 4 или 5; таким образом, вероятность угадывания оказывается 20-25%. Безусловно, это большая величина, поэтому данная конструкция используется лишь в наиболее простых тестовых заданиях, предназначенных для проверки знания каких-то обязательных положений (правил, законов, дат и пр.) [19].

Задания множественного выбора – объединяют в себе n (по числу вариантов ответов) заданий одиночного выбора, каждое из которых независимо от других может быть либо истинным, либо ложным. По сути, в таких заданиях несколько из предложенных вариантов ответа оказываются истинными по отношению к заданному вопросу [19].

Пример

Из перечисленных животных млекопитающими являются:

1. осьминог
2. **дельфин**
3. **гепард**
4. стрекоза
5. попугай
6. варан

Вероятность полного его угадывания равна произведению вероятностей угадывания каждого отдельного вопроса, которая составляет ½. Таким образом,

$$P = \left(\frac{1}{2}\right)^n$$

В классической теории тестирования – только при верных ответах на все вопросы (независимо от доли правильности выполнения – например, если

правильно отвечено на 5 вопросов из 6 (доля выполнения 83%), задание все равно не будет зачтено) [19].

Задания открытого типа – ответ представляет собой определенную последовательность символов (букв, цифр) – слово (несколько слов, предложение), которое при проверке сравнивается со эталонным словом. Задание считается выполненным, если ссылка и введенное слово полностью совпадают. В общем случае, вероятность угадать ответ

$$P = 0$$

Пример

Введите фамилию автора произведения «Война и мир» _______

Рекомендации по составлению заданий открытого типа:

- ответ должен быть однозначным;
- нежелательно использовать в качестве ответа предложение (группу слов), так как это увеличивает вероятность грамматической ошибки (что особенно важно для компьютерногоконтроля);
- если ответ – число, предпочтительно, чтобы оно был целым; если приходится использовать в качестве ответа вещественное число, в эталоне должен быть представлен интервал чисел, при попадании в который ответ будет считаться верным.

Задания соответствия – это установление соответствия между двумя множествами объектов.

Пример

Для физических величин 1-5 укажите единицы их измерения в СИ из а-е:

1) Сила а) Гц

2) Перемещение	b) м/с
3) Частота	c) м
4) Ускорение	d) Н
5) Путь	e) м/с^2
6) Координата	

Ответами такого задания будут пары сочетающихся элементов: 1-d, 2-c, 3-a, 4-e, 5-c, 6-c (дистрактор (b) не используется).

Подобные задания также, по сути, являются заданиями выбора, только выбор в отношении каждого из n элементов первого множества осуществляется из m вариантов ответов, содержащихся во втором множестве. Вероятность угадывания равна [19]

$$P = \left(\frac{1}{m}\right)^n$$

Рекомендации по составлению заданий данного типа:

- количество элементов в каждом множестве – 4-6;
- предпочтительнее, если количество элементов множеств будет различным ($n \neq m$); одинаковое количество элементов сразу настраивает тестируемого на то, что каждому элементу первого множества обязательно соответствует один и только один элемент второго;
- допускается использование в качестве элементов графических объектов (рисунков, схем, формул), обозначенных цифрами или буквами.

Как и в случае с заданиями множественного выбора, задание соответствия считается выполненным, только при 100% правильности ответов.

Задания на установление последовательности – ответом является расставление событий, шагов алгоритма, элементов ряда и т.п. в правильной последовательности.

Пример

Расставьте по времени правления русских императоров династии Романовых:

A. Николай I
B. Александр I
C. Петр III
D. Иван VI
E. Павел I

Ответом является последовательность букв (по сути, эталонное слово) **DCEBA**.

На каждом шаге осуществляется выбор одного верного из оставшихся вариантов ответов. Вероятность полного угадывания всех ответов задания при n элементах в последовательности будет равна [19]

$$P = \frac{1}{n} \cdot \frac{1}{(n-1)} \cdot \ldots \cdot \frac{1}{2}$$

Рекомендации по составлению заданий данного типа:

- желательное количество элементов в последовательности – 5-6 (что заметно снижает вероятность угадывания: при n = 5 P ≈ 0,8%; при n = 6 P ≈ 0,14%);
- последовательность должна быть однозначной (не допускать возможности перестановки каких-либо элементов).

В заключение необходимо еще раз подчеркнуть, что разнообразие форм используемых тестовых заданий обеспечивает их дифференциацию по уровню

сложности, что, в свою очередь, необходимо для дифференциации тестируемых по уровню учебных достижений. К сожалению, это плохо осознается разработчиками образовательных программ – очень часто их материалы представляют собой набор тестов с одним вариантом ответа, которые в лучшем случае представляют собой лишь проверку знаний.

1.3 Модели жизненного цикла программного обеспечения

Разработка любого программного обеспечения обязательно проходит через пять этапов создания готового продукта: анализ требований, проектирование, программирование, тестирование и эксплуатация. Набор этих этапов называется последовательностью фаз жизненного цикла программного обеспечения (ПО). Под жизненным циклом ПО подразумевают ряд событий, происходящих с системой в процессе её создания и дальнейшего использования. Так же жизненный цикл характеризуется временем от начального момента принятия решения о необходимости создания ПО, до момента его ввода в эксплуатацию [5].

Каждый этап жизненного цикла определяется однозначными задачами, методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. Результатами анализа, в частности, являются функциональные и информационные модели, и соответствующие им диаграммы. Жизненный цикл ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах [9].

Структуру, определяющую последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла называют моделью жизненного цикла ПО. Модель зависит от масштаба, сложности проекта и специфики условий, в которых система создается и функционирует. Любая технология базируется на определенных представлениях о жизненном цикле, выстраивает свои методы и инструменты вокруг фаз и

этапов жизненного цикла, именно поэтому необходимо опираться на различные модели при разработке ПО [21].

Подходящий подход к циклу разработки выбирается исходя из требований проекта. В процессе разработки программного обеспечения используются пять основных типов жизненных циклов:

- каскадная;
- V-образная;
- спиральная;
- инкрементная;
- итерационная.

Разберем каждую модель подробнее.

Каскадная модель (англ. waterfall model) – модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки [7].

Рис.7 Каскадная модель [14]

Эта модель редко используется на практике, поскольку требования заказчика могут уточняться, изменяться и дополняться, таким образом, проект или система не будут соответствовать нужным требованиям.

Каскадная модель с промежуточным контролем. Учитывая столь

серьезный недостаток каскадной модели, в неё были добавлены обратные связи с каждым этапом жизненного цикла, что, в свою очередь, увеличивало затраты на разработку в десятки раз, но также позволяло повысить надежность системы в целом. Модифицированная модель получила название каскадная модель с промежуточным контролем или водоворот. Эта модель наиболее предпочтительна в небольших проектах, где требования заранее известны, понятны и четко зафиксированы.

Рис.8. Каскадная модель с промежуточным контролем [14]

V-образная модель. Унаследовала структуру «шаг за шагом» от каскадной модели. V-образная модель применима к системам, которым особенно важно бесперебойное функционирование. Например, прикладные программы в клиниках для наблюдения за пациентами, интегрированное ПО для механизмов управления аварийными подушками безопасности в транспортных средствах и т.д. [12] Особенность модели заключается в том, что она направлена на тщательный анализ и тестирование продукта, который уже находится на ранних этапах проектирования. Этап тестирования проходит одновременно с соответствующим этапом разработки. Например, модульные тесты пишутся во время кодирования. Эта модель используется в малых и средних проектах, где требования четко определены и фиксированы.

Рис.9. V-образная модель [27]

Спиральная модель – модель процесса разработки программного обеспечения, сочетающий в себе как итеративность, так и этапность. Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла [18]. Модель предполагает четыре этапа для каждого витка: планирование, анализ рисков, конструирование, оценка результата и при удовлетворительном качестве переход к новому витку.

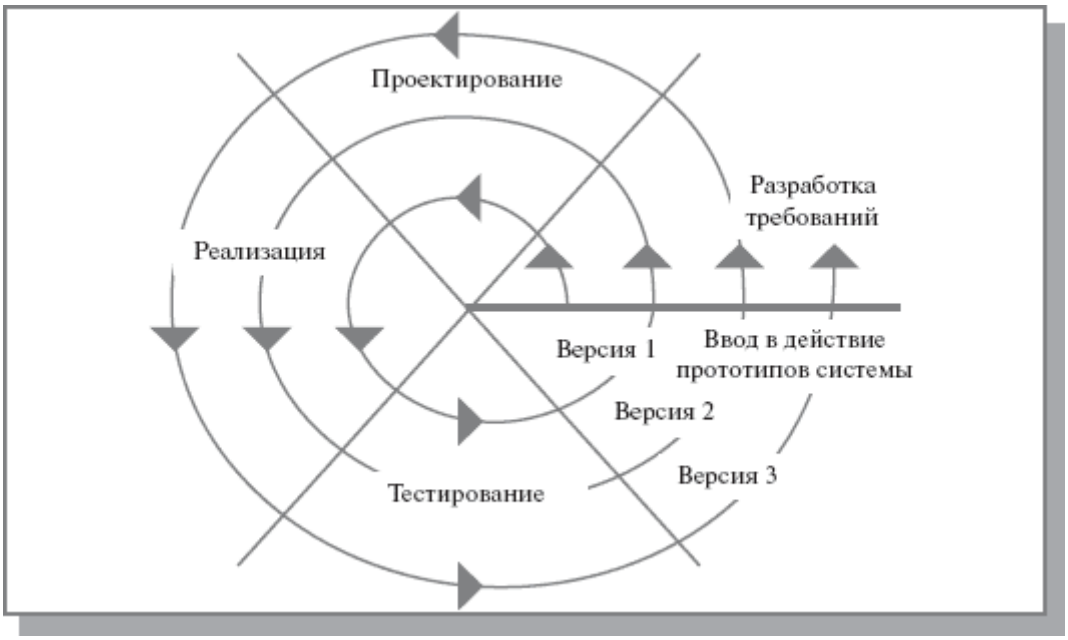


Рис.10. Спиральная модель [14]

Эта модель не подходит для небольших проектов, она разумна для сложных и дорогостоящих проектов, например, таких как разработка системы документооборота большой компании, когда каждый следующий шаг требует больше анализа для оценки последствий, чем программирования.

Инкрементная и итерационная модели. В инкрементной модели (рис. 11) полные требования к системе делятся на различные сборки. Терминология часто используется для описания поэтапной сборки ПО. Имеют место несколько циклов разработки, и вместе они составляют жизненный цикл «мульти-водопад». Цикл разделен на более мелкие легко создаваемые модули. Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования. Процедура разработки по инкрементной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов». Процесс продолжается до тех пор, пока не будет создана полная система. Инкрементные модели используются там, где отдельные запросы на изменение ясны, могут быть легко формализованы и реализованы.

Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований. Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала. Понимая конечную цель, стремятся к ней так, чтобы каждый шаг был результативен, а каждая версия — работоспособна [21].

Рис.11. Инкрементная модель [21]

На рисунке 12 показано итерационная «развитие» Моны Лизы. Как видно, на первой итерации есть лишь набросок Джоконды, на второй итерации появляются цвета, а третья – добавляет деталей, насыщенности и завершает процесс. В инкрементной же модели (рис. 13) функциональность продукта строится по частям, продукт состоит из частей. В отличие от итерационной модели, каждый кусочек представляет собой целостный элемент.

Рис.12. Итерационная модель на примере картины [21]

Рис.13. Инкрементная модель на примере картины [21]

Примером итерационной разработки является распознавание речи. Первые исследования и подготовка научного аппарата начались очень давно, сначала – в мыслях, потом – на бумаге. С каждой новой итерацией качество распознавания улучшалось. Однако до сих пор не удалось добиться идеального распознавания, поэтому проблема до сих пор не решена полностью.

На практике модели жизненных циклов ПО редко используются в чистом виде. В рамках одной или нескольких моделей создаются различные методологии, которые ускоряют процесс разработки и внедрения проекта. Одна из таких методологий – RAD (rapid application development, быстра разработка приложений) которая является разновидностью инкрементной и водопадной модели. Под концепцией RAD подразумевается, что сложный проект разбивается на мини-проекты, что обеспечивает высокую скорость разработки и адаптируемость проекта к меняющимся требованиям.

Технологию RAD применяют, когда выполнение проекта необходимо в сжатые сроки, нечетко определены требования к ПО, возможно разбиение проекта на функциональные компоненты, а также не требуется высокая вычислительная сложность ПО

Этапы разработки включают в себя (рис 14):

1. Планирование – совокупность согласованных требований,

полученных в процессе взаимодействия заказчика и исполнителя

2. Пользовательское проектирование – разработка моделей и прототипов, которые содержат все необходимые функции и которые позволяют заказчику понять, изменить и выбрать рабочую модель, отвечающую всем требованиям.
3. Конструирование – разработка программ и приложений.
4. Переключение – операции по преобразованию данных, тестированию, переходу на новую систему и внедрения в организацию.

Рис.14. Модель быстрой разработки приложений (RAD) [10]

На основании вышеизложенного, стоит сказать, что для решения цели данной выпускной квалификационной работы, подходит модель RAD. Её мы и будем использовать.

1.4 Технологии разработки и шаблоны проектирования веб-приложения

Интернет (Internet) – это глобальная компьютерная сеть, через которую передаются различные данные между компьютерами из любой точки мира. Сеть появилась в 1969 году В США, когда четыре американских университета впервые подключили мэйнфреймы – большие компьютеры.

Интернет включает в себя следующую инфраструктуру: скоростные каналы передачи данных, специальные компьютеры, которые реализуют управление передачей пакетов данных (маршрутизаторов), программное обеспечение, а также большое количество подключенных к ней компьютеров (с серверным и клиентским ПО). Каждому подключенному компьютеру к сети Интернет задается уникальный цифровой адрес, который называется адресом Интернет протокола, или IP адресом. Этот адрес представляет собой целые 4-х байтовые (беззнаковые) числами, которые записываются в виде последовательности значений каждого байта, разделенных символом точка – «.». Например, 109.123.152.2. Программное обеспечение сети Интернет работает по технологии «клиент-сервер», а это значит, что все используемые программы делятся на два типа:

- Серверы – это пассивные программы, которые ожидают запросы от клиентов, оперативно обрабатывают и отправляют запрашиваемую информацию клиентам.
- Клиенты – это активные программы, с которыми, как правило, работает пользователь сети на своем компьютере и которые, отправляют запросы серверам для выполнения некоторой работы (чаще всего получение какой-либо информации) [20].

Для установления взаимодействия между клиентами и серверами необходимо, чтобы они следовали одинаковым наборам правилам описания запросов и ответов на них – протоколам передачи данных. Базовый уровень сети

Интернет работает с двумя основными протоколами – Internet Protocol (IP) и Transmission Control Protocol (TCP), которые часто вместе называются TCP/IP. Они дают возможность передавать данные в виде специально оформленных пакетов ограниченного размера. Работа с протоколами IP и TCP, как правило, поддерживается на уровне операционной системы.

Веб-сеть, или Всемирная паутина – наиболее известный сервис Интернета. Первоначально веб-сеть – World Wide Web, WWW, Web – определялась как информационная система для коллективной работы пользователей, которая предоставляла доступ к гипертекстовым документам в Интернете. Важно отметить, что поскольку этот сервис имеет высокую популярность, нередко под Интернет-сетью (т.е. физической сетью) понимают веб-сеть (т.е. виртуальную сеть).

Сегодня веб-сеть – это платформа, на основе которой разрабатываются различные приложения и целые информационные системы. Взаимосвязь между веб-приложениями и технологиями сети Интернет показана на рис. 15

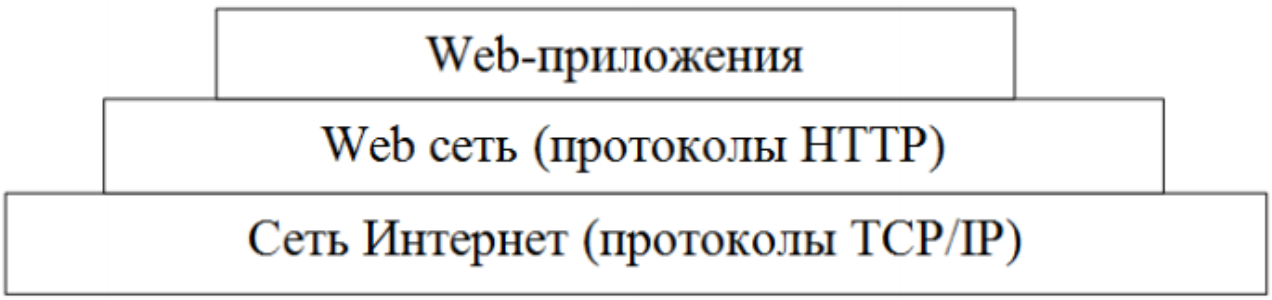


Рис.15. Взаимосвязь веб-приложений с другими технологиями [20]

В 1989 году сотрудник Европейской организации ядерных исследований CERN Тим Бернерс-Ли разработал первую гипертекстовую систему коллективной работы с документами, позволяющую выполнять переходы между ними. Также исследователь разработал основные стандарты, на которых строится данная система. Сегодня Тим Бернерс-Ли является главой Консорциума Всемирной паутины.

Веб-сеть не является физической сетью. Это служба, которая предоставляет пользователям доступ к связанному, с помощью ссылок, набору гипертекстовых документов и веб-приложений. Веб-сеть состоит из набора программного обеспечения (клиентов и серверов) и огромного количества информационных ресурсов, которые работают вместе в соответствии с набором согласованных стандартов. Основные веб-стандарты:

- URL – стандарт задания адресов ресурсов сети;
- HTTP – протокол взаимодействия между клиентами и серверами;
- HTML – язык описания информационных ресурсов (гипертекста);
- CSS – язык форматирования информационных ресурсов – CSS;
- JavaScript – язык выполнения программ на стороне клиента.

В основе веб-сети лежат веб-сайты – website, site – «место», или просто сайты. Веб-сайт – это совокупность логически связанных ресурсов, которые объединены одним адресом (т.е. доменным именем или IP-адресом). В свою очередь, все ресурсы веб-сайтов можно разделить на два типа:

- статические ресурсы – это HTML-документы, изображения, мультимедиа файлы, любые файлы данных, к которым есть доступ;
- динамические ресурсы – это веб-приложения: программные; шаблоны веб-страниц; скрипты; программные объекты и т.п., которые обычно по запросу формируют HTML документы. [20]

Остановимся подробнее на веб-приложения и на технологиях их разработки.

Веб-приложение – это клиент-серверное приложение, которое помогает клиенту взаимодействовать с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, в основном, на сервере, а обмен информацией происходит по сети [4]. То, что клиенты не зависят от конкретной операционной системы

пользователя, является одним из основных преимуществ этого подхода. Именно по этой причине веб-приложения являются кроссплатформенными службами. Вместо того, чтобы писать разные версии для Microsoft Windows, macOS, GNU/Linux, Android, iOS и других операционных систем, приложение создается один раз для произвольно выбранной платформы и на ней разворачивается.

Несколько лет назад веб-сайты были набором статических HTML-страниц, но в процессе перехода к высокоскоростному Интернету технологии создания сайтов быстро изменились, что привело к обширному распространению веб-приложений. Эти веб-сайты содержат интерактивные элементы, инструменты персонализации, а также позволяют обеспечивать взаимодействие между заказчиком и организацией (например, получение заказов или платежей), управлять проектами, планировать и отслеживать задачи бизнеса и дают возможность динамического управления контентом в реальном времени. Как уже было сказано, веб-приложение – это клиент-серверное приложение, в котором клиент работает с сервером с помощью браузера, а веб-сервер отвечает за сервер. Так, клиент, пользуясь веб-браузером, отправляет HTTP-запрос на конкретный URL-адрес, указывающий на какой-либо динамический ресурс, а затем сервер создает HTML-страницу, которая отображается клиенту средствами браузера. Методы разработки веб-приложений делятся на несколько категорий:

- Подходы, основанные на программировании или скриптах: внешние программы или скрипты; расширения веб-сервера;
- Подходы, основанные на использовании шаблонов веб-страниц, включающих вставки кода скриптов и специальных серверных тэгов;
- Объектные среды (каркасы, фреймворки, паттерны) [20].

Несмотря на то, что это категории частично совпадают (а также различные мнения о том, к какой категории относится конкретная технология разработки),

большая часть наиболее популярных подходов связана с одной конкретной категорией. Рассмотрим каждую категорию подробнее.

Программные подходы. Самый простой способ динамически генерировать веб-страницы в ответ на HTTP-запрос – делегировать работу по решению требуемой задачи и формированию HTML-страницы внешней программе, которая получает входные параметры, переданные в HTTP-запросе и создает выходную страницу в HTML. Технология Common Gateway Interface (CGI) стала первой независимой от веб-сервера программной технологией, которая широко использовалась для создания и запуска веб-приложений. Она определяла набор правил, которым должна следовать программа, чтобы она могла выполняться на различных HTTP серверах и операционных системах.

Так, согласно CGI технологии, при поступлении в веб-сервер HTTP-запроса, который включает ссылку не на статическую страницу, а на CGI программу (например, prog.exe), создается новый процесс, в котором запускается требуемая прикладная программа. Технология CGI определяет, каким образом передавать параметры, включенные в HTTP-запрос в такую программу. Передача входных данных может выполняться как с использованием фиксированного набора переменных среды, которые могут создаваться одной программой и использоваться другими, так и через входные данные функции, с которой начинается работа программы, а результаты работы программы (HTML-страница) возвращаются с помощью стандартного потока вывода [8].

Технология CGI является достаточно простым способом динамически формировать информацию в веб-сети. Однако она имеет существенные недостатки, которые делают ее не практичной в большинстве случаев. Основная проблема CGI-приложений заключается в том, что при каждом запросе клиента сервер загружает это приложение в отдельное адресное пространство, а потом инициирует его выполнение и выгрузку. Эта специфика ограничивает производительность приложений и не позволяет одновременно обрабатывать

большое число клиентских запросов [15]. Еще одной особенностью CGI-программ является тот факт, что в этих программах программный код и код набора полностью смешаны. Для того чтобы изменить структуру веб-страниц, дизайнер должен знать программирование и обладать соответствующими навыками.

Попыткой объединения переносимости CGI-приложений с эффективностью их использования стала технология FastCGI. Суть данного подхода заключается в том, что вместо генерации нового процесса для каждого HTTP-запроса, FastCGI не закрывает процессы, связанные с CGI-скриптами, после окончания обработки, а использует их для обработки новых запросов к CGI-программам. Таким образом, процесс инициализируется один раз и в дальнейшем его можно использовать для многократной обработки запросов [21].

Модули сервера, выполняющие набор функций FastCGI, взаимодействуют с HTTP-сервером с помощью собственных API, которые добиваются скрытия деталей реализации и конфигурирования от FastCGI приложений. Однако разработчики все равно должны знать специфику реализации технологии FastCGI, поскольку модули различных типов серверов не совместимы между собой.

Способом преодоления недостатков технологий CGI может быть расширение возможностей веб-серверов с помощью специальных компонентов. Использование таких расширений позволит программам, которые формируют HTTP-ответы, выполняться более эффективно, без необходимости их завершения после обработки каждого запроса и за счет использования общих ресурсов несколькими приложениями. Эти технологии, как правило, позволяют хранить в основной памяти данные сеансов работы пользователей, которые взаимодействуют с приложением в течение большого количества HTTP-запросов [8].

Одной из таких технологий расширения архитектуры веб-сервера является интерфейс Java Servlet API, связывающий веб-сервер с виртуальной машиной Java Virtual Machine (JVM). Она входит в состав платформы Jakarta EE, ранее известной как Java Enterprise Edition. Виртуальная машина JVM поддерживает выполнение специальной Java-программы (контейнер сервлетов), отвечающей за управление данными сеанса работы и выполнения Java-сервлетов. Сервлеты – это специальные классы на языке Java (программы), имеющие доступ к информации из HTTP-запросов и формирующие HTTP-ответы, которые возвращаются браузерам. Контейнер сервлетов (т.е. среда выполнения) отвечает за получение от веб-сервера HTTP-запросов на выполнение сервлетов [21].

Подходы на основе шаблонов. Они используют в качестве адресуемых объектов (имеющий URL адрес) не программы или скрипты, а «шаблоны». Шаблоны – это HTML-файлы с дополнительными «тэгами» (серверные, которое используются только на стороне сервера), задающими методы включения динамически формируемого контента. Файл шаблона содержит HTML-код, который описывает общую структуру страницы, и дополнительные серверные тэги, размещенные так, чтобы формируемой с их помощью содержание странице имело требуемый вид [20].

Active Server Pages (ASP), Java Server Pages (JSP) и PHP относят к наиболее распространенным технологиям разработки веб-приложений на основе шаблонов.

Active Server Pages (ASP) – это разработанная Microsoft технология создания веб-приложений, использующая объектную модель интерфейса, созданного на основе ISAPI-фильтра. ASP упростила задачи генерации HTML-страниц и дала возможность производить обращение к компонентам баз данных. Принцип интерфейса приложения, заключается в том, что код, фрагменты которого есть на веб-странице, интерпретируется веб-сервером и предоставляет пользователю готовый результат выполнения выбранных фрагментов кода [21].

Технология ASP, объединенная с бесплатным веб-сервером Internet Information Server (IIS), быстро набрала популярность среди программистов, использующих Visual Basic, которые оценили возможность использования в шаблонах языка VBScript.

Java Server Pages (JSP) – это технология создания веб-приложений, которая основана на однократной компиляции Java-кода (сервлета) при первом обращении к нему с последующим выполнением методов этого сервлета и помещением полученных результатов в набор данных, которые затем отправляются в браузер [21]. Технология JSP была ответом компании Sun на популярность технологии Microsoft ASP.

PHP: Hypertext Preprocessor разработан в 1994 году Расмусом Лердорфом набор CGI-скриптов, который превратился в полноценный скриптовый язык программирования [26]. PHP представляет собой сценарный язык программирования и программное средство для создания веб-страниц. Эта технология позволяет решать множество задач, однако главной областью использования является написание PHP-скриптов, работающих на стороне сервера. В PHP входит CGI-интерфейс, интерпретатор языка, а также набор функций для доступа к базам данных и различным объектам Глобальной сети Интернет [21].

Подходы на основе объектных сред. Несмотря на то, что обычные скриптовые технологии на стороне сервера используют различные объекты, они не позволяют разрабатывать и использовать собственные классы и создавать на их основе новые объекты. Поэтому дальнейшее развитие веб-технологий было связано с созданием специальных объектно-ориентированных технологий разработки веб-приложений. Использование данных технологий позволяет сделать разработку веб-приложений сходной с разработкой обычных объектно-ориентированного программного обеспечения.

Объектные среды, или фреймворки, являются следующим уровенем совершенствования разработки веб-приложений. Вместо объединения разметки и логики в единый модуль, объектные среды поддерживают принцип отделения содержания от представления. Модули ответственные за создание контента отделяется от модулей, которые показывают это содержание в конкретном формате.

Отделение содержания от представления является важным в связи с тем, что увеличивается гибкость приложения (возможность его изменения с небольшими затратами); улучшается разделение ответственности между веб-дизайнерами и программистами.

В настоящее время есть два подхода к созданию объектно-ориентированных веб-приложений:

- подходы, основанные на наборе специальных веб-страниц (веб-форм), связанных с описаниями классов, объекты которых будут создаваться и использоваться при их вызове (например, технология ASP.Net Web Forms или технология JavaServer Faces);
- подходы, основанные на использовании наборов классов, соответствующих шаблону Model-View-Controller (MVC) (например, технологии на основе языка Java – Tapestry, Struts, Spring, на основе языка PHP – Lavarel и технология компании Microsoft – ASP.NET MVC) [20].

Подробнее остановимся на подходе разработки веб-приложений, соответствующему шаблону MVC.

Учитывая цель снижения трудозатрат на разработку сложного ПО, предположим, что необходимо использовать готовые унифицированные решения. Ведь шаблон действий облегчает общение между разработчиками, позволяет ссылаться на известные конструкции и снижает количество ошибок.

Как уже говорилось, большинство фреймворков веб-приложений в настоящее время реализуют шаблон проектирования MVC, но используются и другие, такие как Model-View-Presenter (MVP) или Model-View-View Model (MVVM).

Использование шаблона Модель-Представление-Контроллер (MVC) означает, что бизнес-логика (модель) отделена от ее визуализации (представления) (рис. 16). Такое разделение увеличивает возможность повторного использования кода, и изменения в одном из компонентов как можно меньше влияют на другие компоненты.

Рис.16. Архитектура MVC [13]

Модель предоставляет данные и реагирует на команды Контроллера, изменяя своё состояние. Представление отвечает за отображение данных Модели пользователю, в ответ на изменения Модели. Контроллер интерпретирует действия пользователя, уведомляя модель о необходимости изменений. Почти все вышеперечисленные ранее фреймворки поддерживают или дополняют данную архитектуру.

Шаблон проектирования Model-View-Presenter (MVP, Модель-Представление-Представитель) является производным от MVC и в основном используется для создания пользовательского интерфейса. Элемент Presenter берет на себя посредническую функцию и отвечает за управление событиями пользовательского интерфейса (такими как наведение курсора мыши). MVP был разработан, чтобы упростить автоматическое модульное тестирование и улучшить разделение проблем в логике отображения.

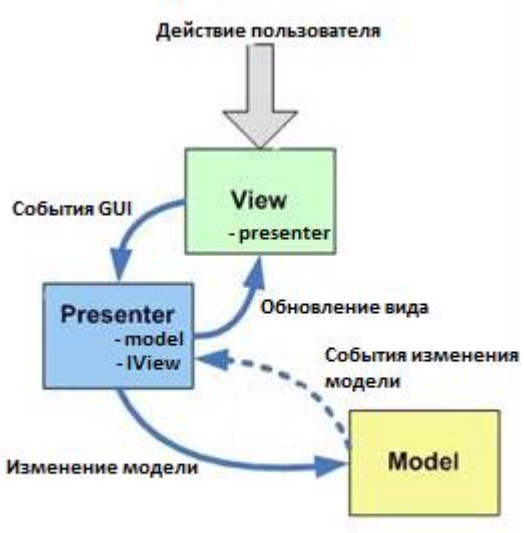


Рис.17. Архитектура MVP [13]

Модель хранит бизнес-логику, представление реализует отображение данных из модели, а представитель реализует взаимодействие между моделью и представлением. Схема MVP похожа на модель MVC и отличается только тем, что между моделью и представлением нет прямых связей, а вместо Контроллера используется Представитель.

Шаблон Model-View-View Model (MVVM, Модель-Представление-Модель представления) используется при проектировании архитектуры приложения. Использование MVVM вместо классического MVC удобно, когда существует связь между пользовательским интерфейсом приложения и бизнес-логикой на вашей платформе разработки.

Модель, как и в MVC, представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения. В этом случае представление представляет собой графический интерфейс, то есть окно, кнопки и т. д. Модель представления является абстракцией представления и предоставляет оболочку данных из Модели.

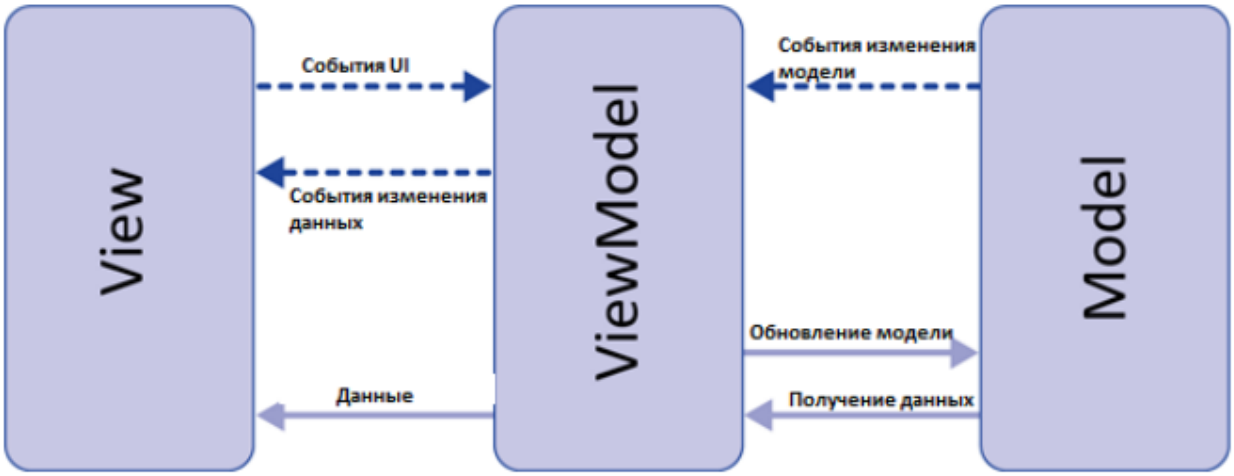


Рис.18. Архитектура MVVM [13]

В качестве технологии разработки веб-приложений за основу выбран язык PHP и концепция MVC. На PHP будет разработан собственный движок, реализующий архитектуру Model-View-Controller, без использования сторонних фреймоворков.

Глава 2. Практическая часть

2.1 Структура веб-приложения

Веб-приложение развернуто на арендованном интернет-хостинге. Для удобного доступа арендован домен второго уровня. Приложение доступно по адресу: https://mob-test.xyz/

Пакетная структура веб-приложения представлена на рисунке 19.

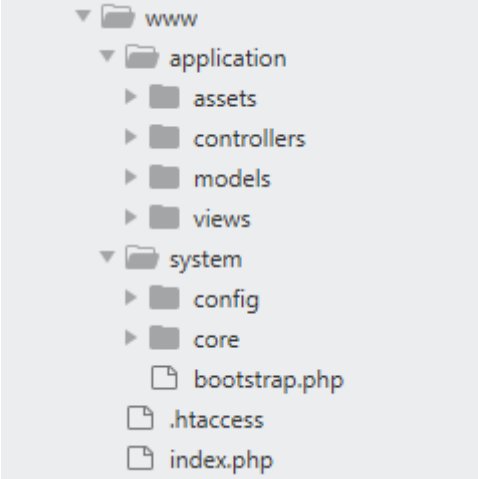


Рис.19. Пакетная структура приложения

Руководствуясь моделью MVC, большинство классов приложения разделены на три категории – models, views и controllers. Находятся они в папках www/application/models/, www/application/controllers и www/application/views соответственно. В папке www/system/core хранятся корневые классы приложения, от которых наследуется большинство используемых классов.

В папке www/core/config хранятся файлы конфигурации, а в папке www/application/assets – стили css, подключаемые скрипты JavaScript, а также изображения, используемые в оформлении ресурса.

Хранение информации о тестах реализовано подключением удаленной базы данных. База данных MySQL расположена на арендованном интернет-

хостинге. В ней созданы три таблицы: ‘users’, ‘right_answers’, ‘users_answers’. Структуру таблиц, вы можете увидеть на рисунках, представленных ниже:

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	login	varchar(255)	utf8_general_ci		Нет	*Нет*		
3	password	varchar(255)	utf8_general_ci		Нет	*Нет*		

Рис.20. Структура таблицы ‘users’

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	LOGIN	varchar(255)	utf8_general_ci		Нет	*Нет*		
3	CODE	varchar(4)	utf8_general_ci		Нет	*Нет*		
4	Q1	varchar(255)	utf8_general_ci		Нет	*Нет*		
5	Q2	varchar(255)	utf8_general_ci		Нет	*Нет*		
6	Q3	varchar(255)	utf8_general_ci		Нет	*Нет*		
7	Q4	varchar(255)	utf8_general_ci		Нет	*Нет*		
8	Q5	varchar(255)	utf8_general_ci		Нет	*Нет*		

Рис.21. Структура таблицы ‘right_answers’

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	CODE	varchar(4)	utf8_general_ci		Нет	*Нет*		
3	USERNAME	varchar(255)	utf8_general_ci		Нет	*Нет*		
4	Q1	varchar(255)	utf8_general_ci		Нет	*Нет*		
5	Q2	varchar(255)	utf8_general_ci		Нет	*Нет*		
6	Q3	varchar(255)	utf8_unicode_ci		Нет	*Нет*		
7	Q4	varchar(255)	utf8_general_ci		Нет	*Нет*		
8	Q5	varchar(255)	utf8_general_ci		Нет	*Нет*		
9	RESULT	varchar(255)	utf8_general_ci		Нет	*Нет*		

Рис.22. Структура таблицы ‘users_answers’

Также, стоит упомянуть, что реализация поддержки различных диагоналей экранов мобильных устройств реализована с помощью подключаемого файла стилей CSS. Было взято простое свободно распространяемое решение из Интернета и интегрировано в веб-приложение.

2.2 Основные методы работы веб-приложения

Файл .htaccess – это файл конфигурации веб-сервера Apache, распространенного решения в Интернете. В данном приложении, его «внутренности» имеют следующий вид:

```
AddDefaultCharset utf-8
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule .* index.php [L]
```

AddDefaultCharset utf-8 – указывает браузеру, что в приложении используется кодировка UTF-8.

RewriteEngine On – директива, включающая режим работы механизма преобразования полученного URL-адреса из адресной строки браузера. Директива RewriteCond определяет условия этого механизма преобразования. Параметры %{REQUEST_FILENAME} !-f и %{REQUEST_FILENAME} !-d вместе с данной директивой, определяют, что если в адресной строке URL-адрес ссылается на какой-либо файл или папку, которая есть на сервере – механизм преобразования применятся не будет. Само же правило RewriteRule .* index.php [L] – преобразует URL вида mob-test.xyz в mob-test.xyz/index.php. Это поможет запустить наше веб-приложение.

Единая точка входа в приложение – это файл index.php:

```
<?php
	require_once 'system/bootstrap.php';
```

Функция require_once вызывает и запускает наш файл загрузки bootstrap.php. Её отличительная черта, от include_once, что при отсутствие вызываемого файла – приложение не запускается, что помогает при выявлении ошибок. Критически важно использовать её, а не include, при вызове системных классов и библиотек.

Уже в bootstrap.php запускаются корневые классы нашего приложения.

```
<?php
	require_once 'system/core/model.php';
	require_once 'system/core/controller.php';
	require_once 'system/core/view.php';
	require_once 'system/core/database.php';
	require_once 'system/core/session.php';
	require_once 'system/config/paths.php';
	require_once 'system/core/router.php';


	Router::start();
```

Рассмотрим каждый из них немного подробнее.

Файл model.php корневой класс для всех моделей приложения. Имеет следующий вид:

```
<?php
class Model
{
	public function __construct() {
		Session::init();
		$this->db = new Database();
	}
}
```

В конструкторе класса запускается функция функция init() класса Session. Её мы рассмотрим позже. Также создается экземпляр класса Database, для работы с базой данных MySQL.

Файл controller.php – корневой класс для всех контроллеров приложения:

```
<?php
class Controller {
	public $model;
	public $view;
	//конструктор
	function __construct()
	{
		$this->view = new View();
		session::init();
	}
}
```

В нём создаются две публичные переменные, которые будут использоваться в потомках этого класса. В конструкторе создается экземпляр класса View, и инициализируется сессия пользователя.

Файл view.php – основной класс для представлений:

```
<?php
class View {
	public function generate($content_view, $template_view, $title = 
'Онлайн-тестирование | mob-test.xyz', $data = null, $header = 'mob-
test.xyz')
	{
		include 'application/views/'.$template_view;

	}
}
```

Метод класса generate используется в контроллерах приложения для вызова представления. Он вызывает так называемый шаблон представления – HTML-каркас с подключенными к нему js-скриптами и css-стилями, внутри которого уже вызывается сам класс представления. Метод имеет следующие параметры:

- $content_view – файл представления;
- $template_view – файл шаблона;
- $title – переменная использующаяся в шаблоне внутри тега <title>, для вывода информации заголовка в адресную строку браузера;
- $data – переменная использующая для передачи данных представлению;
- $header – переменная использующаяся в шаблоне внутри тега <header>, для вывода информации в шапку веб-страницы.

database.php – класс для создания подключения к БД MySQL. Имеет следующий вид:

```
<?php
  class Database extends PDO
```

```
  {
   public function __construct() {
          require_once 'system/config/paths.php';
     parent::__construct(DB_TYPE.':host='.DB_HOST.';charset=UTF8'.';dbna
me='.DB_NAME, DB_USER, DB_PASS);
   }
 }
```

Наследуется от системного класса PDO – класса для работы с SQL-запросами. С помощью родительского конструктора создает соединение с БД. Использует константы из файла paths.php:

```
<?php
     define('URL', 'https:// mob-test.xyz'/');
     define('forTitle', ' | mob-test.xyz');

     define('DB_TYPE', 'mysql');
     define('DB_HOST', 'spl92.hosting.reg.ru');
     define('DB_NAME', 'u1293251_MobTest');
     define('DB_USER', 'u1293_admin');
     define('DB_PASS', '*********’);
```

Первые две константы используются для сокращения кода при выполнении рутинных операции перехода или отображения информации в заголовке. Остальные – используются в подключении к БД.

Файл session.php – описывает класс, функции которого работают с сессией и cookie пользователя. Имеет следующий вид:

```
<?php
class Session
{
     public static function init()
     {
          session_start();
     }

     public static function set($key, $value) {
          $_SESSION[$key] = $value;
     }

     public static function get($key) {
          if(isset($_SESSION[$key]))
               return $_SESSION[$key];
```

```
	}

	public static function destroy() {
		unset($_SESSION);
		session_destroy();
	}

	public static function getUser() {
		if(isset($_COOKIE['authUser']))
			return $_COOKIE['authUser'];
	}
}
```

- метод init() вызывает функцию session_start(), которая создает или возобновляет сессию пользователя;
- метод set() устанавливает идентификатор для текущей сессии;
- метод get() позволяет проверить, существует ли сессия с запрашиваемым идентификатором;
- метод destroy() – очищает идентификатор сессии и закрывает сессию;
- метод getUser() позволяет узнать имя авторизованного пользователя из данных cookie. Использует в модели авторизации.

Последние две строки файла bootstrap.php подключают файл router.php и запускают в нем метод start().

router.php – маршрутизатор нашего веб-приложения. Все перемещения по директориям и ссылкам реализуются с помощью него. Основная концепция заключается в следующей структуре:

Ссылка вида mob-test.xyz/main/index/parameters, запустит контроллер **main**, а в нём выполнится его метод **index** с аргументами **parameters**. Если представить описанную ссылку как массив с разделителем в виде косой черты, то второй элемент массива всегда будет являться контроллером, третий элемент – методом этого контроллера или «экшеном», четвертый – аргументами для «экшена».

Теперь рассмотрим подробно как это реализовано в коде:

```
<?php
class Router
{
	public static function start(){

		//задаем значения контроллера и его экшена стартовой страницы
		$controller_name = 'main';
		$action_name = 'index';


		//делаем массив из ссылки в адресной строке
		$routes = explode('/', $_SERVER['REQUEST_URI']);

		//получаем имя контроллера (если оно есть)
		if ( !empty($routes[1]) )
		{
			$controller_name = $routes[1];
		}

		// получаем имя экшена
		if ( !empty($routes[2]) )
		{
			$action_name = $routes[2];
		}

		// добавляем префиксы
		$model_name = 'model_'.$controller_name;
		$controller_name = 'controller_'.$controller_name;
		$action_name = 'action_'.$action_name;

		// подгружаем файл с классом модели (файла модели может и не
быть)
		$model_file = strtolower($model_name).'.php';
		$model_path = "application/models/".$model_file;
		if(file_exists($model_path))
		{
			include "application/models/".$model_file;
		}

		// подгружаем файл с классом контроллера
		$controller_file = strtolower($controller_name).'.php';
		$controller_path =
"application/controllers/".$controller_file;
		if(file_exists($controller_path))
		{
			include "application/controllers/".$controller_file;
		}
```

```
            else
            {
                  Router::ErrorPage404();
            }

            // создаем экземляр контроллера
            $controller = new $controller_name;
            $action = $action_name;



//проверяем ссылку на наличие аргументов вызываем действие контроллера
            if(method_exists($controller, $action))
            {
                  if ( !empty($routes[1]) ){
                        $controller->$action($routes[3]);
                  }
                  else {
                        $controller->$action();
                  }
            }
            else
            {
                  Router::ErrorPage404();
            }
      }

      function ErrorPage404()
      {
            header('Location:../404');
     }
}
```

Основные пояснения по работе метода start() даны в коде в виде комментариев. Отдельно хочу отметить, что при отсутствии в полученной ссылке метода или аргументов (третьего или четвертого элементов массива), для метода устанавливается значение action_index – значение по умолчанию, на отрисовку веб-страницы. Аргументов же – может и не быть. А при отсутствии в ссылке контроллера (второго элемента массива), переменой $controller_name присваивается значение главной страницы приложения.

Также, обрабатываются исключения. Если указанный в ссылке контроллер или его метод не существует – произойдет редирект на страницу

«404». Метод ErrorPage404() с помощью функции header() отправляет HTTP-заголовок, который перенаправляет браузер по адресу mob-test.xyz/404 – метод Router::start() запускается вновь.

Теперь рассмотрим работу контроллера, на пример класса controller_main. Код файла www/application/controller/controller_main.php выглядит так:

```
<?php
class Controller_main extends Controller
{
	function __construct()
	{
		parent::__construct();
		$this->model = new model_main();
	}

	public function action_index()
	{
		$this->view->generate('main_view.php', 'template_view.php');
	}

	public function action_auth()
	{
		$this->model->auth();
	}

	public function action_error()
	{
		$this->view->generate('main_view.php', 'template_view.php',
'Онлайн-тестирование'.forTitle, 'Теста с таким кодом - не существует');

	}
  }
```

У этого класса, есть конструктор. По мимо того, что в нем создается экземпляр класса model_main (модели main), также объявляется конструктор родительского класса controller. В нём, как было описано раньше, создается экземпляр класса view и создается/возобновляется сессия пользователя.

- метод action_index() нам уже знаком. Он подгружает шаблон template_view.php, внутри которого инициализируется представление главной страницы – main_view.php;
- метод action_auth() вызывается при обработке HTML-формы внутри main_view$ и выполняет метод auth() в экземпляре класса model_main;
- метод action_error() – заново перерисовывает представление main, но уже передает аргументе $data, чтобы оповестить об ошибке.

Давайте теперь рассмотрим представление main. Сама стартовая страница выглядит так:

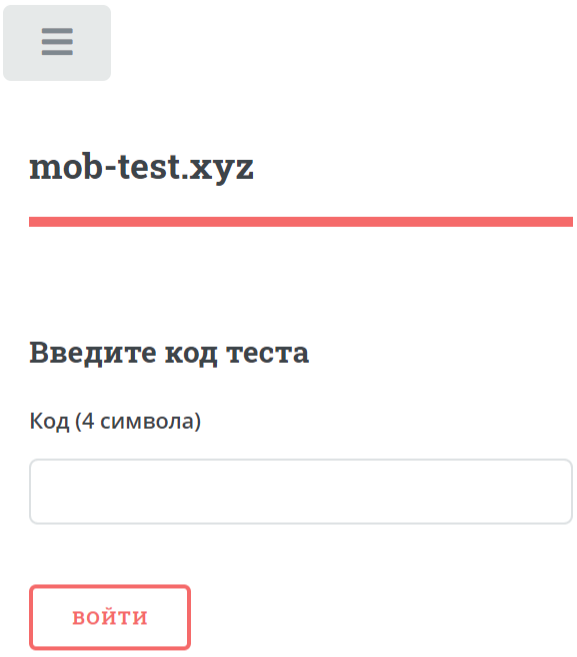


Рис.23. Скриншот страницы mob-test.xyz/main

А файл www/application/view/main_view.php выглядит так:

```
<h3> <?php echo $data ?> </h3>
<h3>Введите код теста</h3>
<form  action="../main/auth" method="POST">
  <label>Код (4 символа)</label><input type="text" name="code"
maxlength="4" minlength="4" required autocomplete="off"><br>
  <label></label><input type="submit" value="Войти">
</form>
```

Как уже было ранее сказано, представление загружается внутри шаблона – «каркаса» из HTML, CSS и JS. Само представление довольно простое – два тега <h3> для отображения информации и форма с <input type=”text”> и кнопкой «Войти».

В одном из <h3> тегов есть вставка php-кода – в данном случае мы вызываем данные из переменной $data, переданной нам из контроллера controller_main.

Параметр action=”..//main/auth” тега <form>, при нажатии на кнопку «Войти», отправит нас по адресу mob-test.xyz/main/auth для выполнения метода action_auth() контроллера controller_main (все недостающие префиксы добавить метод Router::start()). Параметр method=”POST” означает, что данные введенные в строку, после нажатия кнопки «Войти», мы сможем получить в глобальной переменной $_POST['code’] (‘code’ из-за параметра name=’code’), а также эти данные не конкатенируют с адресной строкой при переходе (в отличии от параметра method=”GET”).

Перейдем к моделям. Модели, по самой концепции шаблона MVC, это набор методов, которые вызываются из контроллера, чтобы получить от модели данные и передать их представлению. Рассмотрим некоторые методы:

```
public function auth() {
    $authUser = $_POST['login'];
    $sth = $this->db->prepare("SELECT id FROM users WHERE login = :login
AND password = MD5(:password)");
    $sth->execute(array(
      ':login' => $authUser,
```

```
      ':password' => $_POST['password']
    ));
    $count = $sth->rowCount();
    if($count > 0)
    {
      session::set($authUser, true);
      setcookie('authUser', $authUser, time()+3600, '/');
      header('Location: ../dashboard');
    }
    else
    {
    header('Location: ../login/error');
    }
  }
```

Это метод auth() класса model_login – метод для авторизации создателей тестов. Опишем подробнее выполняемый здесь код, связанные с запросом к БД:

- `$authUser = $_POST['login'];` - создаем переменную $authUser и добавляем в неё данные из формы с name=”login”;
- `$sth = $this->db->prepare("SELECT id FROM users WHERE login = :login AND password = MD5(:password)");` - создаем переменную $sth для подготовленного запроса. Обращаемся к нашему экземпляру класса PDO (созданного в конструкторе родительского класса Model) и передаем ему необходимую строку с SQL запросом, пока что заменяя необходимые нам данные на, так называемые, плейсхолдеры (например :login). Обратите внимание, что плейсхолдер :password мы будет передавать в хешированном виде, потому что в таком виде он хранится в БД;
- `$sth->execute(array(':login' => $authUser,':password' => $_POST['password']));` - выполняем SQL-запрос, меняя плейсхолдеры на необходимые нам значения, и получаем результирующие данные затронутых строк БД;
- `$count = $sth->rowCount();` - записываем количество затронутых строк в $count.

Дальше, нам нужно проверить, нашлась ли строка с таким логин и паролем в БД (if($count > 0)). Если нет, то запускаем метод action_error() класса controller_login. Если да, то:

- `session::set($authUser, true);` – устанавливаем индитификатор этой сессии равным имени авторизовавшегося пользователя;
- `setcookie('authUser', $authUser, time()+3600, '/');` - создаем cookie-данные $_COOKIE=$authUser, хранимые 1 час и доступные из любого каталога веб-ресурса;
- `header('Location: ../dashboard');` - запускаем контроллер controller_dashboard.

Рассмотрим еще метод getResultTest() класса model_testing.

```
public function getResultTest($user_answers, $right_answers)
{
	$count_right_answers = 0;
	foreach ($user_answers as $k => $v)
		{
			if (array_key_exists($k, $right_answers))
				{
					$var = strcasecmp($user_answers[$k], 
$right_answers[$k]);
					if ($var == 0)
					{
						$count_right_answers = 
$count_right_answers + 1;
					}
				}
			}

		$user_answers['RESULT'] = 
$count_right_answers.'/'.count($right_answers);

	return $user_answers;
}
```

Данный метод сравнивает ответы после прохождения теста с ключами теста и добавляет данные о результате в массив ответа, для дальнейшей его отправки в БД.

Входные аргументы метода - $user_answer, $right_answer. Это собственно массивы данных. Первый с ответами пользователя прошедшего теста, второй – это массив с правильными ответами на этот тест. Такой массив получается при выполнении метода getRightAnswers() этого же класса model_testing.

Так как, данные в массивах различаются количеством элементов (но необходимые элементы у нас имеют одинаковые идентификаторы в обоих массивах) мы не можешь сравнить их напрямую.

Функция foreach($user_answers as $k => $v) позволяет перебрать каждый элемент массива $user_answers, не забывая про идентификатор элемента. Это позволяет сравнить элементы двух массивов, если их идентификаторы совпадают.

С помощью функции array_key_exists($k, $right_answers) мы проверяем, есть ли в массиве $right_answers идентификатор элемента $k (идентификатора одного из перебираемых элементов массива $users_answers). Если мы нашли схожие идентификаторы – сравниваем их функцией сравнения строк без учета регистра strcasecmp($user_answers[$k], $right_answers[$k]). Возвращаемое значение функции равно 0 или строки равны. В таком случае увеличиваем наш счетчик правильных ответов на 1. В конце, осталось только добавить количество правильных ответов в массив $user_answers и вернуть этот массив вызываемой стороне.

Рассмотрим также программное решение, по отображению HTML-таблицы с данными из массива с помощью php. Данное решение встроено в представление (dashboard_check_test_view.php), отображающее результаты тестируемых, по определённому тесту. Данный код выводит правильные ответы на тест:

```
<?php
echo '<div class="table-wrapper"> <table class="alt">';
echo '<tr>';
foreach ($data[1] as $k => $v)
```

```
{
      echo '<td><b>'.$k.'</td>';

}
echo '</tr>';
echo '<tr>';
foreach ($data[1] as $k => $v)
{
      echo '<td>'.$v.'</td>';

}
echo '<tr>';
echo '</table></div>';
?>
```

С помощью функции echo(), можно выводить HTML-теги, позволяющие отрисовать таблицу с помощью тегов <table>,<tr>,<td>.

В начале, функцией echo() мы выводим открывающие теги нашей будущей таблицы и открывающий тег строки таблицы. После этого, функцией foreache(), мы перебираем данные массива полученного от контроллера controller_dashboard. Массив $data[1] одномерный, поэтому в первую строку таблицы выводим идентификаторы элементов массива, добавляю возле них открывающий и закрывающий тег <td> - столбца таблицы. Вывод в следующей строке уже элементов массива , а не их идентификаторов тривиален. Обращаю внимание на правильную подстановку функции echo() с необходимыми для отрисовки тегами HTML. Результат изображен на рисунке 24.

Правильные ответы теста

Q1	Q2	Q3	Q4	Q5
2	23	толстой	431533	54213

Рис.24. HTML-таблица с данными из массива $data[1]

2.3 Инструкция по работе с веб-приложением

Для перехода на тест, необходимо на стартовой странице указать код теста и нажать кнопку «Войти» (присутствует демо-тест с кодом QWER):

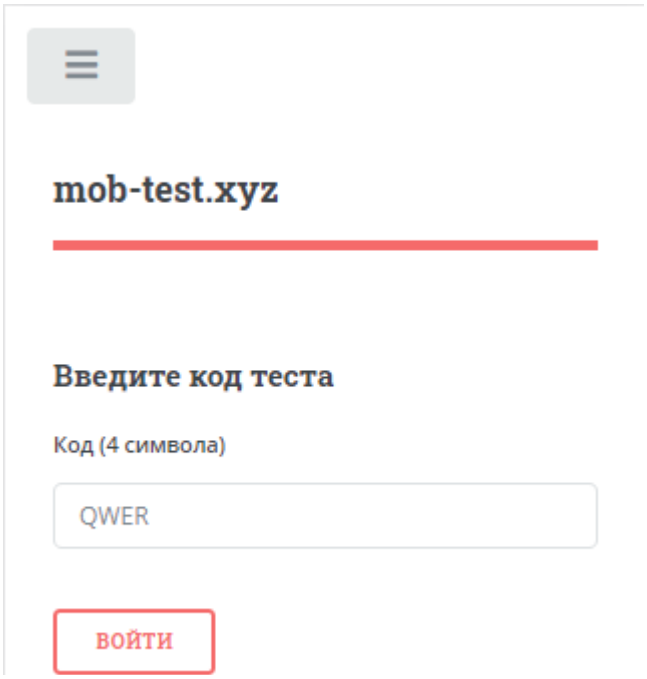


Рис.25. Страница авторизации для прохождения тестирования

После выполнения заданий теста, для проверки результата и отправки его на сервер, необходимо нажать кнопку «Отправить результат». Также присутствует кнопка «Сбросить», которое очищает все чекбоксы и поля для ввода (рис.26). После прохождения теста, учащийся будет перекинут на страницу с его результатом (рис.27).

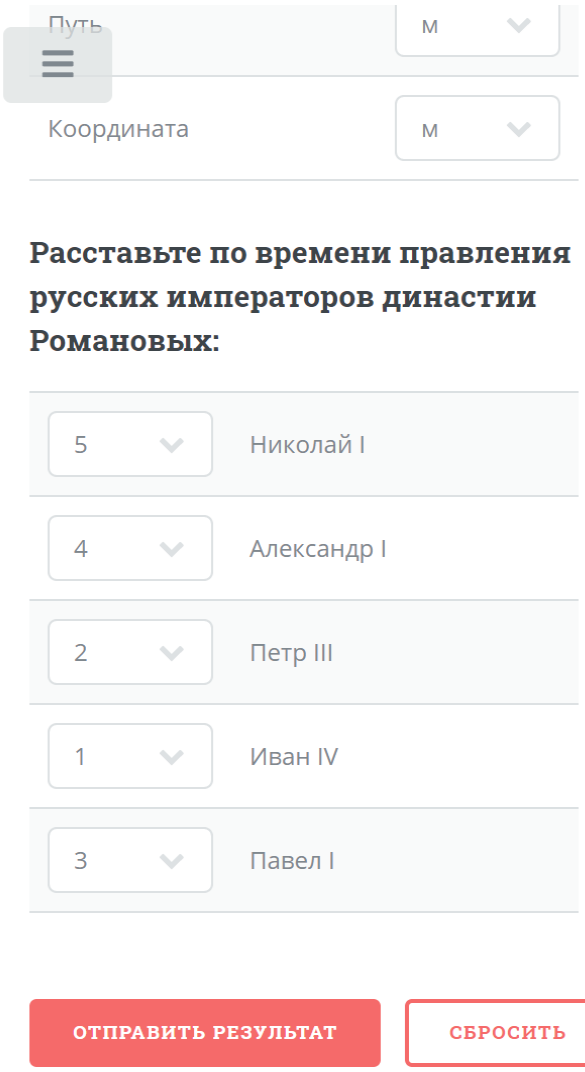


Рис.26. Страница тестирования

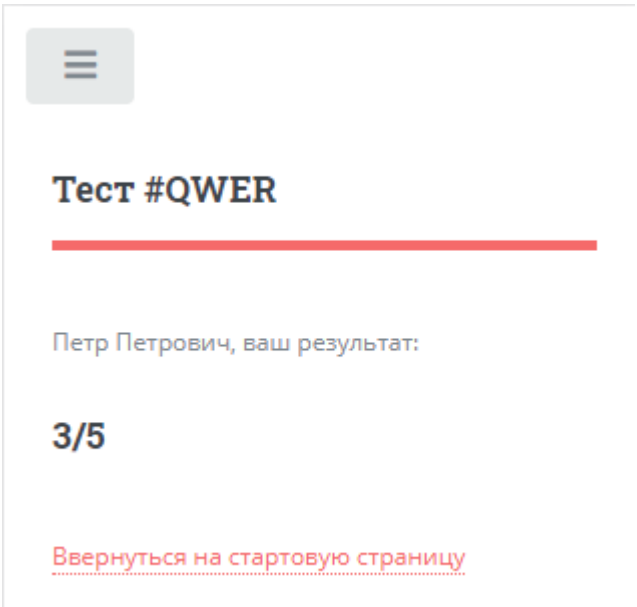


Рис.27. Страница результатов тестирования

Авторизация создателя теста, происходит по нажатию ссылки «Авторизация», в левом верхнем углу открывающегося бокового меню, так называемого «гамбургера».

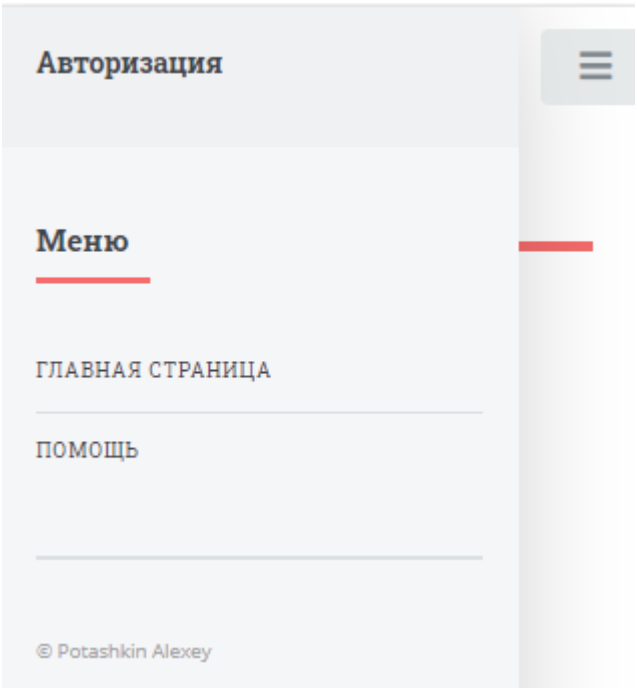


Рис.28. Боковое меню

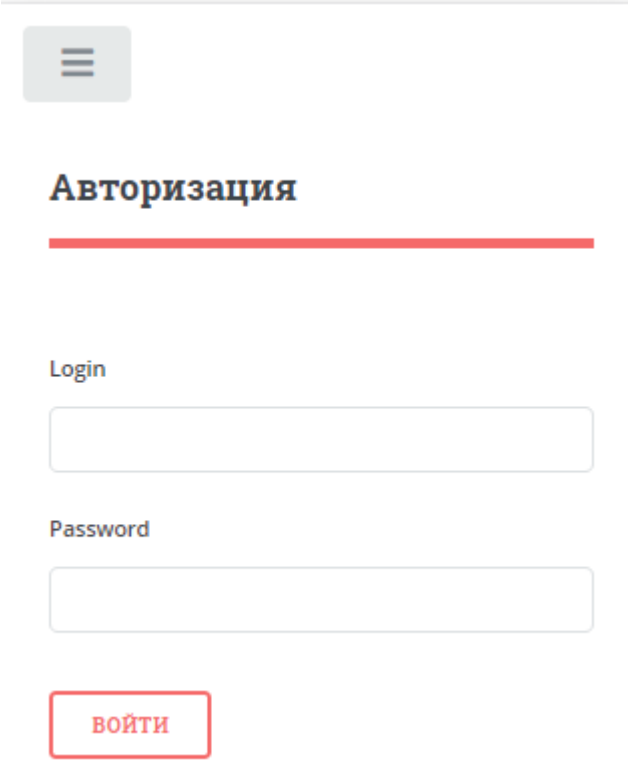
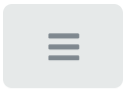


Рис.29. Страница авторизации для создателей тестов

Существует тестовый пользователей с логином/паролем – user/test. Ему доступен просмотр результатов демо-теста «QWER». Сама страница с результатами тестирования представлена на рисунке 30.



Результаты теста QWER

Правильные ответы теста

Q1	Q2	Q3	Q4	Q5
2	23	толстой	431533	54213

Ответы тестируемых

USERNAME	Q1	Q2	Q3	Q4	Q5	RESULT
Поташкин Алексей	2	23	толстой	423415	23313	3/5
Иван Иванов	0	23	пушкин	431533	54213	3/5
Алина	2	23	Толстой	451534	00010	3/5
Абдурахман ибн Хоттаб	2	23	Толстой	431533	54213	5/5
Петр Петрович	2	1456	ЛЕРМОНТОВ	431533	54213	3/5

Рис.30. Страница с результатами теста «QWER»

2.4 Апробация

Апробация разработанного веб-приложения для школьного онлайн-тестирования проводилась методом экспертных оценок. В качестве экспертов выступали студенты Уральского государственного педагогического университета в количестве 9 человек. Им была предоставлена ссылка на сервис, объяснен функционал, дан код для прохождения анкетирования на сайте. Анкета разделена на 3 части – в первой части задаются вопросы с градацией (вопросы 1 и 2), во второй части – закрытые вопросы (вопросы 3 и 4), в третьей – открытый вопрос (вопрос 5).

Анкета:

1. По шкале от 1 до 5, где «1» – это «очень **не**удобно», а «5» – «очень удобно», оцените удобство прохождения тестирования с мобильного устройства, на сайте mob-test.xyz.
2. По шкале от 1 до 5, где «1» – это «очень **не**удобно», а «5» – «очень удобно», оцените удобство создания нового теста, на сайте mob-test.xyz.
3. Возникали ли ошибки или баги при работе с сайтом?
    - да;
    - нет;
    - затрудняюсь ответить.
4. Выбрали ли вы данный сайт для онлайн-тестирования среди школьников?
    - да;
    - нет;
    - затрудняюсь ответить.
5. Что необходимо добавить или изменить для повышения удобства работы с сайтом?

При ответе на первый вопрос, 56% экспертов отметили, что проходить онлайн-тестирование с мобильного устройства, на сайте mob-test.xyz, «очень удобно». Ещё 33% - что «удобно». Среднее значение ответов – 4,44, где где «1» – это «очень неудобно», а «5» – «очень удобно».

В ответе на второй вопрос, 67% анкетированных отметили, что создавать новые тесты на сайте – «удобно» или «очень удобно». Среднее значение ответов – 4,11.

8 из 9 экспертов не обнаружили ошибок или багов при работе с сайтом. Лишь один эксперт в третьем вопросе отметил, что затрудняется ответить.

67% процентов опрошенных, готовы выбрать сайт mob-test.xyz для онлайн-тестирования, среди школьников.

Последний вопрос, являлся открытым. Три эксперта заявили о том, что необходимо реализовать добавление мультимедийных файлов (изображений, видео и т.д.) при создании теста. Ещё один эксперт, пожелал увидеть функционал доступа к тесту по QR-коду. Получив обратную связь от экспертов, в будущем, возможно продолжить работу над увеличением функционала веб-приложения.

Подробные результаты отображены в приложении 1 данной выпускной квалификационной работы.

Большинство опрошенных экспертов, считаю веб-приложения удобным для проведения онлайн-тестирования с мобильных устройств и готовы использовать его при работе с школьниками. Подавляющее большинство не обнаружило ошибок или багов в приложении, мешающих работе с ним. Таким образом разработку сервиса школьного онлайн-тестирования для мобильных устройств можно считать успешной.

Заключение

В связи с усовершенствованием компьютерных технологий стали разрабатывать различные системы для записи алгоритмов, которые назвали языками программирования. Языки программирования содержат в себе набор вычислительных формул, который, в процессе работы, переводят в алгоритм.

Выполнение выпускной квалификационной работы позволило закрепить ранее изученный материал по дисциплинам «Языки и технологии программирования», «Теория и методики обучения информатики», «Технологии компьютерного тестирования» изучить новый для меня материал.

В ходе выполнения данной выпускной квалификационной работы на основе различных источников данных была проанализирована заданная предметная область.

В результате выполнения выпускной квалификационной работы была полностью спроектировано и реализовано веб-приложение для школьного онлайн-тестирования. В процессе тестирования ошибок не обнаружено.

Цель выпускной квалификационной работы достигнута, задачи выполнены.

В заключении стоит отметить, что разработанная система актуальна для преподавателей. Целесообразно было бы продолжить работу над добавлением дополнительного функционала.

Список литературы

1. Аванесов В.С. Композиция тестовых заданий. 2 изд. – М.: Центр тестирования, 2002. – 239 с. Аванесов В.С. http://testolog.narod.ru (дата обращения: 19.01.2021).
2. Алмазова, С.Л. Методы психологической диагностики [Текст]: учеб. пособие / С.Л. Алмазова; Урал. гос. пед. ун-т. – Екатеринбург, 2010. – 116 с.
3. Бесплатный конструктор тестов – создать тест онлайн – URL: https://konstruktortestov.ru (дата обращения 21.02.2021)
4. Веб-приложение. – URL: https://ru.wikipedia.org/wiki/Веб-приложение (дата обращения 25.01.2021)
5. Зараменских Е. П. Управление жизненным циклом информационных систем: монография. / Е. П. Зараменских. – Новосибирск: Издательство ЦРНС, 2014. – 270 с.
6. Ингенкамп К. Педагогическая диагностика. – М.: Педагогика, 1991. – 240 с.
7. Каскадная модель. – URL: https://ru.wikipedia.org/wiki/Каскадная_одель (дата обращения 23.01.2021).
8. Лобода Ю. Г., Орлова О. Ю. Технологии разработки веб-приложений. / Ю. Г. Лобода, О. Ю. Орлова // Научные работы. Одесская национальная академия пищевых технологий. – Одесса, Национальная библиотека Украины имени В. И. Вернадского, 2014. – Вып. 46, том 1. – С. 239-244.
9. Модели жизненного цикла программного обеспечения. – URL: https://habrahabr.ru/post/111674/ (дата обращения: 22.01.2021).
10. Модель жизненного цикла RAD. – URL: https://commons.wikimedia.org/w/index.php?curid=17833563 (дата обращения 23.01.2021)
11. Новиков А.М., Новиков Д.А. Методология. – М.: СИН-ТЕГ. – 668 с.
12. Орлик С. В. Введение в программную инженерию и управление жизненным циклом ПО. / С. В. Орлик. – Москва: Символ-Плюс, 2005. – 183 с.
13. Паттерны для новичков: MVC vs MVP vs MVVM. – URL: https://habr.com/post/215605/ (Дата обращения 09.02.2021).
14. Проектирование информационных систем. Структура и модели жизненного цикла. – URL: http://daxnow.narod.ru/index/0-13 (дата обращения: 23.01.2021).

15. Сенина А. А. Обзор основных современных технологий разработки веб-приложений. / А. А. Сенина // XIII Всероссийская научно-практическая конференция «Технологии Microsoft в теории и практике программирования». Технологии разработки и проектирования информационных систем. – Томск, ТНИПУ, 2010. – Секция 6. – С. 233-235.
16. Сидоров С.В. Педагогическая диагностика [Электронный ресурс] // Сидоров С.В. Сайт педагога-исследователя – URL: http://si-sv.com/publ/1/pedagogicheskaja_diagnostika/14-1-0-538 (дата обращения: 19.01.2021).
17. Система управления обучением. – URL: https://ru.wikipedia.org/wiki/Система_управления_обучением (дата обращения 02.02.2021)
18. Спиральная модель. – URL: https://ru.wikipedia.org/wiki/Спиральная_модель (дата обращения 23.01.2021).
19. Стариченко Б.Е., Мамонтова М.Ю., Слепухин А.В. Методика использования информационно-коммуникационных технологий в учебном процессе. Ч. 3. Компьютерные технологии диагностики учебных достижений. Учебное пособие [Текст] / Под ред. Б.Е. Стариченко / Урал. гос. пед. ун-т. Екатеринбург, 2014. – 179 с.
20. Тузовский А.Ф. Проектирование Интернет приложений / А.Ф. Тузовский; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2010. – 200с.
21. Хусаинова К.И. Проектирование и разработка программного обеспечения для информационного сопровождения деятельности студенческого профсоюзного комитета ВУЗа / ПГГПУ, Пермь, 2018. – 126 с.
22. Челышкова М.Б. Теория и практика конструирования педагогических тестов: Учебное пособие. – М.: Логос, 2002. – 432 с.
23. INDIGO – Программа создания тестов и тестирования сотрудников и студентов – URL: https://indigotech.ru (дата обращения 21.02.2021)
24. LearningApps.org – создание мультимедийных интерактивных упражнений – URL: https://learningapps.org (дата обращения 21.02.2021)
25. Moodle – Open-source learning platform | Moodle.org – URL: https://moodle.org (дата обращения 21.02.2021)
26. PHP. – URL: https://ru.wikipedia.org/wiki/PHP (дата обращения 13.12.2020)

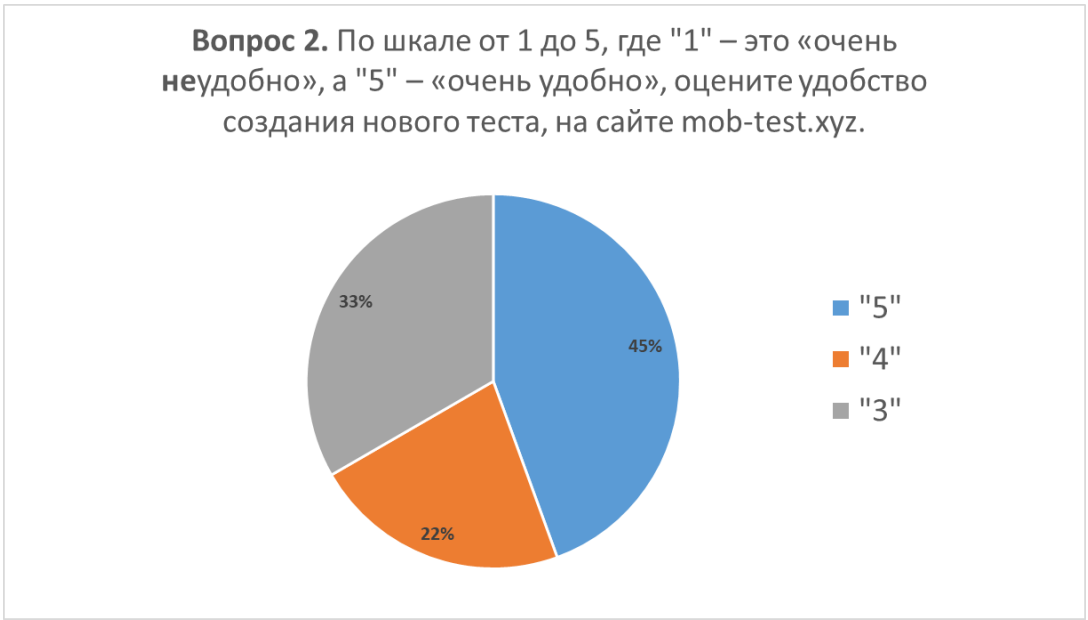
27. V-Model. – URL: https://ru.wikipedia.org/wiki/V-Model (дата обращения 23.01.2021).

Приложение 1

Результаты апробации.

Результаты первой части вопросов анкеты:

Часть 1	Эксперты									Среднее значение
	№1	№2	№3	№4	№5	№6	№7	№8	№9	
Вопрос 1	5	4	5	4	4	5	3	5	5	4,44
Вопрос 2	5	3	5	3	4	5	3	5	4	4,11



Результаты второй части вопросов анкеты:

Часть 2	Эксперты								
	№1	№2	№3	№4	№5	№6	№7	№8	№9
Вопрос 3	нет	нет	нет	нет	нет	нет	нет	затрудняюсь ответить	нет
Вопрос 4	да	затрудняюсь ответить	да	затрудняюсь ответить	да	да	нет	да	да

Результаты третьей части вопросов анкеты:

Часть 3. Эксперты	Вопрос 5. Что необходимо добавить или изменить для повышения удобства работы с сайтом?
№1	-
№2	"возможно использовать видео/аудио материал в тесте"
№3	"ничего"
№4	"возможность добавлять мультимедия в режими создания теста"
№5	-
№6	-
№7	"необходима возможность добавлять картинки в тест"
№8	-
№9	"чтобы к тесту можно было подключиться по QR-коду"